

Dr. Dobb's Journal of

#113 MARCH 1986
\$2.95 (3.95 CANADA)

Software Tools

FOR THE PROFESSIONAL PROGRAMMER

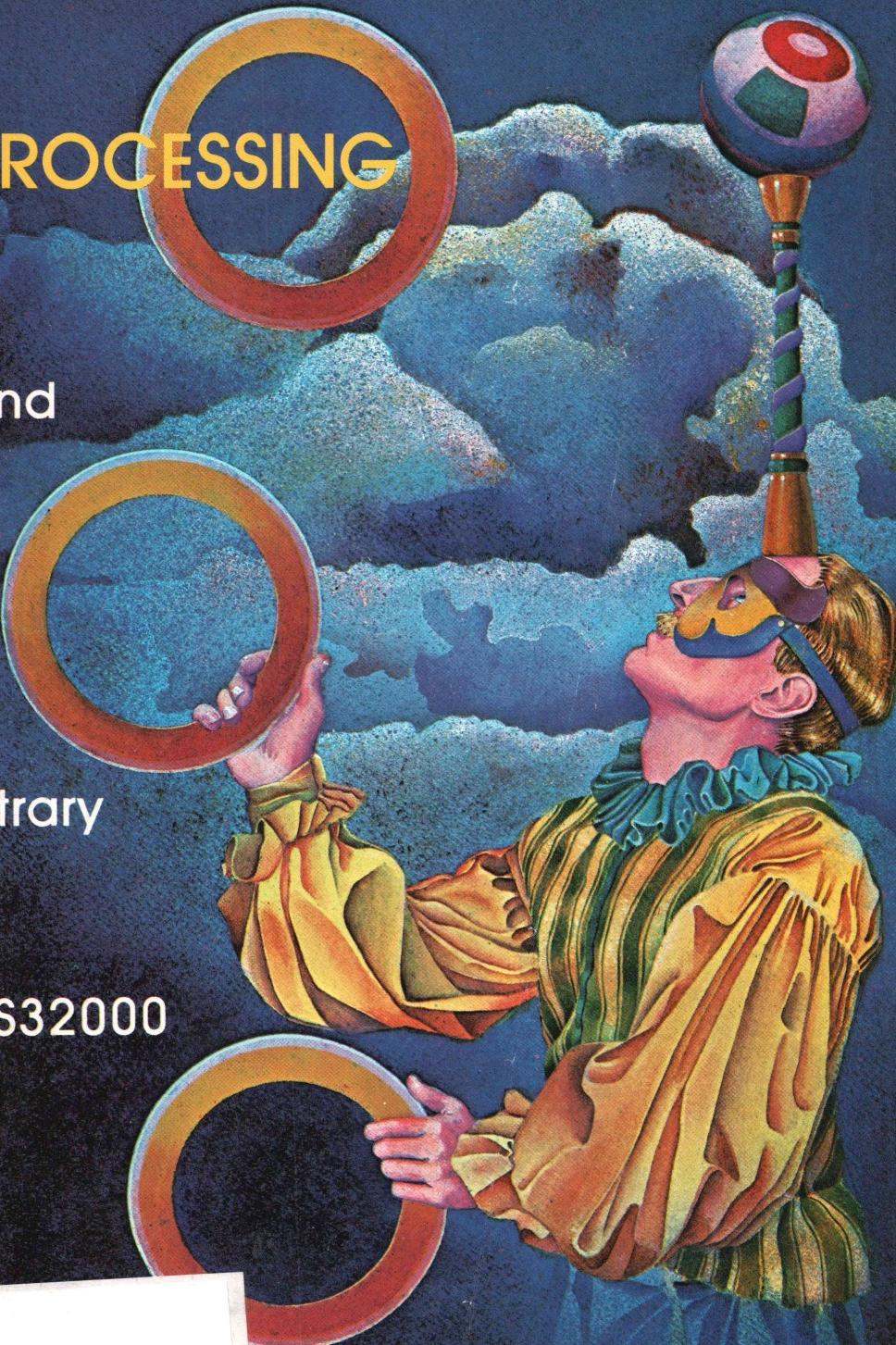
PARALLEL PROCESSING

Concurrency and
Turbo Pascal

What makes
DOS fast?

Minimizing arbitrary
functions

MC68000 vs. NS32000





SOME HISTORIC BREAKTHROUGHS DON'T TAKE AS MUCH EXPLAINING AS COMPUERVE.

But then, some historic breakthroughs could only take you from the cave to the tar pits and back again.

CompuServe, on the other hand, makes a considerably more civilized contribution to life.

It turns the personal computer into something useful.

CompuServe is an information service. Just subscribe, and 24 hours a day, 7 days a week, a universe of information, entertainment and communications is at your service.

A few of the hundreds of things you can do with CompuServe:

COMMUNICATE

Easyplex™ Electronic Mail puts friends, relatives and business associates in constant, convenient touch.

CB Simulator lets thousands of enthusiastic subscribers "chatter away" on 72 different channels.

Over 100 Forums welcome you to join their online "discussions." They're for everyone from computer owners and gourmet cooks to physicians and game players.

Bulletin Boards let you "post" messages where thousands will see them.

HAVE FUN

Our full range of games includes "You Guessed It!" the first online TV-style game show played for real prizes; *Mega-Wars III*, the ultimate in interactive excitement; board; parlor; sports and educational games.

SHOP

THE ELECTRONIC MALL™ gives you 'round the clock shopping for name brand goods and services at discount prices from nationally known stores and businesses.

SAVE ON TRIPS

TWA Travelshopper™ lets you scan schedules and fares, find the best bargains and order tickets online.

A to Z Travel/News Service provides latest travel news plus complete information on over 20,000 hotels worldwide.

MAKE PHI BETA KAPPA

Grolier's Academic American Encyclopedia's Electronic Edition is a complete, constantly updated general reference encyclopedia.

The College Board, operated by the College Entrance Examination Board, helps you prepare for the SAT, choose a college and get financial aid.

BE INFORMED

The AP News Wire (covering all 50 states and the nation), the Washington Post, USA TODAY Update and business and trade publications are constantly available. And our electronic clipping service lets us find, clip and file specific news for reading at your convenience.

INVEST WISELY

Comprehensive Investment Help includes complete statistics on over 10,000 NYSE, AMEX and OTC securities. Historic trading statistics on over 50,000 stocks, bonds, funds, issues and options. Five years of daily commodity quotes. Standard & Poor's Value Line. And over a dozen other investment tools.

Site II provides demographic and sales potential information by state, county and zip code for the entire country.

And now for the pleasant surprise.

Although CompuServe makes the most of any computer, it's a remarkable value. You get low start-up costs, low usage charges and local-phone-call access in most major metropolitan areas.

Here's how to use CompuServe.

CompuServe is "menu-driven," so beginners can simply read the lists of options on their screens and then type in their selections.

Experts can just type in "GO" followed by the abbreviation for whatever topic they're after.

In case of confusion, typing "H" for help brings immediate instructions.

And you can ask general questions either online through our free Feedback service or by phoning our Customer Service Department.

How to subscribe.

To access CompuServe, you'll need a CompuServe Subscription Kit; a computer, terminal or communicating word processor; a modem and in some cases, easy-to-use communications software.

With your Subscription Kit, you'll receive a \$25 usage credit, a complete hardcover Users Guide, your own exclusive user ID number and preliminary password, and a subscription to CompuServe's monthly magazine, *Online Today*.

Subscription Kits are available in computer stores, electronic equipment outlets, retail stores and catalogs. You can also subscribe with materials you'll find packed right in with many computers and modems sold today.

Make a move of historic proportions. Subscribe to CompuServe today.

To receive our free informative brochure or to order direct, call or write:

CompuServe®

Information Services
P.O. Box 20212, 5000 Arlington Centre Blvd.
Columbus, OH 43220

800-848-8199

In Ohio, call 614-457-0802



Optotech, Inc.

The 5½ inch Optical Disk Drive Is Here!

Optical Disk Drive 5984

- *200 megabytes on a removable cartridge.*
- *Fast access read and write.*
- *For PCs and minis.*
- *Extensive interface software.*
- *Available Now.*

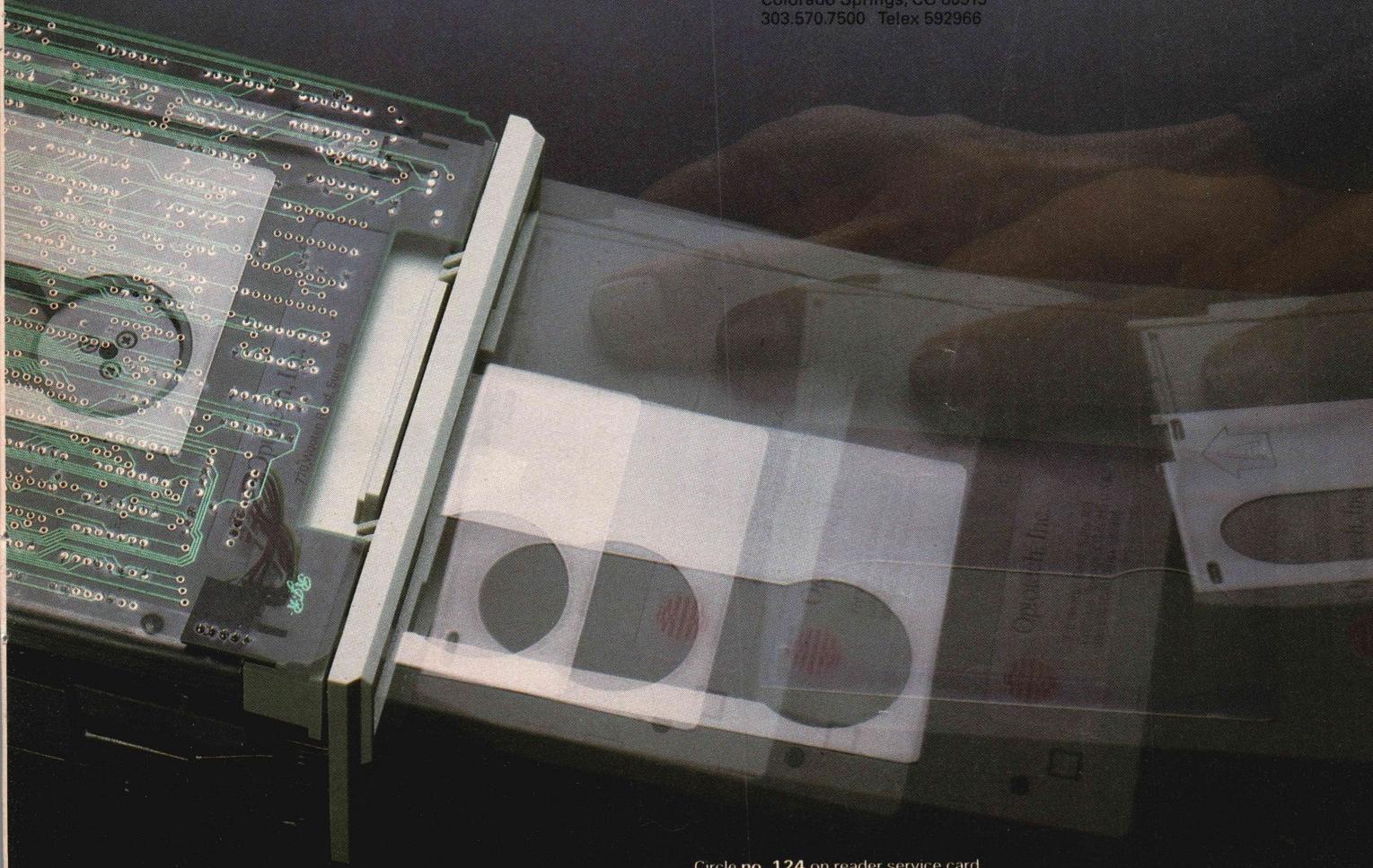
The Preferred Solution For:

- *Database—large, portable, indelible and updatable*
- *Online Mass Storage—integral backup*
- *Imaging—capacity with removability*



Optotech, Inc.

770 Wooten Road
Colorado Springs, CO 80915
303.570.7500 Telex 592966





The Best C Book

A Powerful C Compiler

One Great C Value \$39.95

A good C book just isn't complete without a good C compiler to go with it. That's why we give you both. You get a comprehensive 450 page book and a full feature standard K&R C compiler with the Unix V7 Extensions. The Book is loaded with examples that teach you how to program in C. And our fast one pass C compiler comes with an equally fast

linker so you don't waste a lot of time watching your disk drives spin. You also get a Unix compatible function library that contains more than 150 functions (C source code included). And if all that isn't enough, we offer you a 30 day money back guarantee. So what are you waiting for? The exciting world of C is just one free phone call away.

Language Features

- Data Types: char, short, int, unsigned, long, float, double
- Data Classes: auto, extern, static, register
- Typedef, Struct, Union, Bit Fields, Enumerations
- Structure Assignment, Passing/Returning Structures

abs	conbuf	feof	getcseg	isascii	movmem	replace	strcat
asm	conc	ferror	getdseg	iscntrl	open	repmem	strcnop
asmx	cos	fflush	getd	isdigit	outp	rewind	strcpy
atan	cprstr	fgets	putd	islower	peek	right\$	strlen
atof	creat	fileno	getdate	isprint	perror	rindex	strncat
atoi	cursblk	filetrap	gettime	ispunct	poke	rmdir	strncmp
atol	curslin	find	geti	isspace	poscurs	scanf	strncpy
bdos	curscol	floor	puti	isupper	pow	setbuf	strsave
bdosx	cursrow	fopen	getkey	itoa	printf	setbufsiz	system
bios	cursoff	fprintf	getmode	keypress	putc	setcolor	tolower
biosx	cursor	fpmts	setmode	left\$	putchar	setdate	toupper
callcx	delete	fread	gets	len	puts	settime	ungetc
ceil	drand	free	getw	log	putw	setjmp	unlink
cfree	exec	freopen	heapsiz	log10	rand	setmem	write
chain	exec1	fscanf	heaptrap	longjmp	read	sin	writechs
character	execv	fseek	hypot	lseek	readatr	sound	xmembeg
chdir	exit	ftell	index	malloc	reach	sprintf	xmemend
chmod	exitmsg	fwrite	inp	alloc	writech	sqr	xmemget
clearerr	exp	getc	insert	mathtrap	readdot	srand	xmemput
close	fabs	getch	iofilter	mid\$	writedot	sscanf	xmemvmem
clrscrn	fclose	putch	isalnum	mkdir	realloc	stacksiz	_exit
cmpstr	fdopen	getchar	isalpha	modf	rename	str	

Functions

MIX Editor \$29.95

When you're programming in a high level language you need a high powered editor. That's why we created a programmable full/split screen text processor. It lets you split the screen horizontally or vertically and edit two files at once. You can move text back and forth between two windows. You can also create your own macro commands from an assortment of over

100 predefined commands. The editor comes configured so that it works just like Wordstar but you can change it if you prefer a different keyboard layout. The editor is a great companion to our C compiler. Because they work so well together we want you to have both. To make sure you do, we're offering the editor for just \$15 when purchased with the C compiler.

ASM Utility \$10

The ASM utility disk allows you to link object files created by Microsoft's MASM or M80 assemblers. Lots of useful assembly language functions are included as examples.

ORDERS ONLY
1-800-523-9520
IN TEXAS
1-800-622-4070

Canadian Distributor
Saraguay Software: 416-923-1500

NOT COPY PROTECTED

Editor	\$ _____	(29.95)
C	\$ _____	(39.95)
C & Editor	\$ _____	(54.95)
ASM Utility	\$ _____	(10.00)
TX Residents	\$ _____	(6.125% sales tax)
Shipping	\$ _____	(see below)
Total	\$ _____	
<input type="checkbox"/> Check	<input type="checkbox"/> Money Order	
<input type="checkbox"/> MC/Visa#	Exp _____	
Shipping Charges: (No charge for ASM Utility)		
USA:	\$5.00/Order	
Canada:	\$10.00/Order	
Overseas:	\$10.00/Editor	• \$20.00/C • \$30.00/C & Editor

<input type="checkbox"/> PCDOS/MSDOS (2.0 or later)	Name _____
<input type="checkbox"/> IBM PC Single Side	Street _____
<input type="checkbox"/> IBM PC Double Side	City _____
<input type="checkbox"/> Tandy 2000	State _____
<input type="checkbox"/> 8 Inch	Zip _____
<input type="checkbox"/> Other _____	Country _____
<input type="checkbox"/> CPM 80 (2.2 or later)	Phone _____
<input type="checkbox"/> 8 Inch	
<input type="checkbox"/> Kaypro II	
<input type="checkbox"/> Kaypro 4	
<input type="checkbox"/> Apple (Z80)	
<input type="checkbox"/> Osborne I SD	
<input type="checkbox"/> Osborne I DD	
<input type="checkbox"/> Morrow MD II	
<input type="checkbox"/> Other _____	

MIX
software
2116 E. Arapaho
Suite 363
Richardson, TX 75081
(214) 783-6001

Ask about our volume discounts.

Dr. Dobb's Journal of

Software Tools

ARTICLES

**Warping space
for minimum
benefits**

**Teaching tasks
to share**

**If it's Tuesday,
this must be
Turbo.**

**A new column
on assembly
language**

**What's up on
the DDJ SIG**

MATH: A Variable Metric Minimizer

by Joe Marasco

Unconstrained minimization seeks the minimum of a function regardless of its form: polynomial, transcendental, or weird.

PASCAL: Concurrency and Turbo Pascal

by Ernest E. Bergmann

Coroutines are an important feature of Modula-2 and Ada; here is a straightforward approach to implementing coroutines in Turbo Pascal.

**FUTURE PROGRAMMING: The Problems of
Parallelism**

by Michael Swaine

Will the future be parallel?

MS DOS: Speeding MS DOS Execution

by Gregg Weissman

Inspired by Dave Cortesi's puzzles regarding MS DOS execution speed, the author interrogated the operating system for the solution.

**PORABILITY: Automatic Porting Between
Pascal Dialects**

by Michael J. Sorens

This article presents a proven technique for automating the task of porting your programs from one Pascal to another.

**PORABILITY: COM: An 8080 Simulator
for the MC68000**

by Jim Cathey

Listing of Cathey's program continued from January

24

36

40

44

50

108

14

122

FORUM**PROGRAMMERS'
SERVICES****EDITORIAL: Sharing**

by Michael Swaine

LETTERS: Comment

by you

CARTOON: Parallel

Processing

by Rand Renfroe

6

OF INTEREST: New

products of interest

to programmers

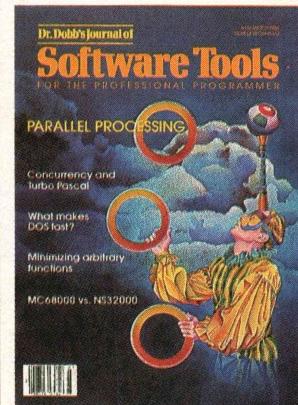
ADVERTISER INDEX:

Where to find

that ad

124

128

**About the Cover**

Parallel processing. Like a juggler whose right hand is better off not knowing what the left is doing, one starts the various processes going and trusts the rhythm of the task to stave off disaster. The cover illustration was done by Tim Gault of Gault Design and Illustration.

This Issue

The advent of parallel processing could force us to rethink the way we solve problems. Mathematical algorithms in general are routinely, almost by implicit definition, sequential. Now, things are changing. Is some new Knuth writing the book on parallel algorithms? Herein, some perspective on parallel processing today and an example of concurrency in Pascal. Also please note the new column on assembly language and Joe Marasco's monumental minimizer.

Next Issue

Our annual look at the techniques of artificial intelligence will present code in Prolog, LISP, and Expert-2, led off by Bob Brown's BRIE (Boca Raton Inference Engine).

It's amazing what you can reveal when you strip.

Introducing a shape that's about to turn on an entire industry.

The Softstrip™ data strip. From Cauzin.

This new technology allows text, graphics, and data to be encoded on a strip of paper, then easily entered into your computer using a scanning device called the Cauzin Softstrip™ System Reader.

Creating a simple, reliable and cost efficient way to distribute and retrieve information.

Softstrip data strips, like those you see here, can contain anything that can be put on magnetic disks.

Facts. Figures. Software programs.

Video games. Product demonstrations.

Sheet music.



The Cauzin Softstrip System Reader is now compatible with the IBM PC, Apple II and Macintosh.

A single strip can hold up to 5500 bytes of encoded data.

It can stand up to wrinkles, scratches, ink marks, even coffee stains.

And it can be entered into your computer with a higher degree of reliability than most magnetic media.



The Cauzin Softstrip System Reader replaces tedious typing by scanning the strip and reading it into your computer.

Simply by plugging the Cauzin Reader into your serial or cassette port and placing it over the strip.

The reader scans the strip, converts it to computer code, and feeds it into any standard communication interface.

Because strips are so easy to generate, most of your favorite magazines and books will soon be using them in addition to long lists of program code.

And you'll be able to enter programs without typing a single line.

There is also software for you to generate your own strips.

Letting you send everything from correspondence to business information using our new technology.

Find out how much you can reveal by stripping. Just take this ad to your computer dealer for a demonstration of the Cauzin Softstrip System Reader.

Or for more information and the name of the dealer nearest you, call Cauzin at 1-800-533-7323. In Connecticut, call 573-0150.

Softstrip™
COMPUTER READABLE PRINT

Cauzin Systems, Inc.
835 South Main St., Waterbury, CT 06706

THE ADVANTAGES OF TURBO PASCAL AND DATA STRIPS

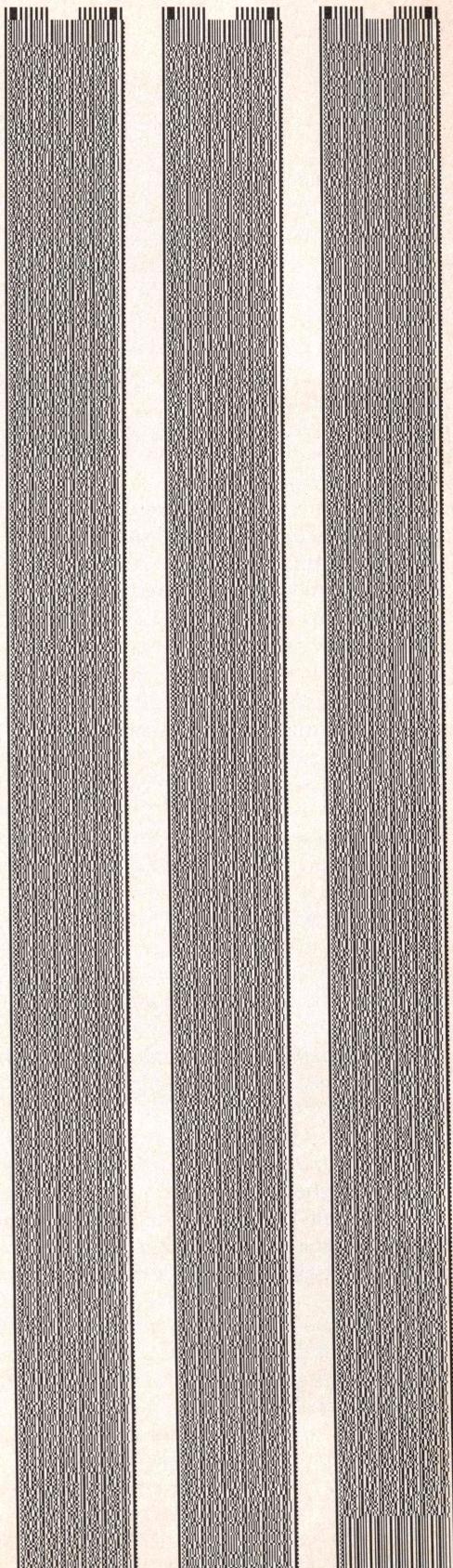
The Cauzin Softstrip™ data strips, on the right, contain the problem described in the article "Concurrency and Turbo Pascal" in this issue.

As you'll recall, Ernest Bergmann, after much trial and error, came up with a means to use Turbo Pascal (CP/M-80) for implementing several concurrent tasks. The techniques are useful for writing software that has to respond to several real-time events, such as a modem program.

As Mr. Bergmann points out: because of the ease of writing and trying variations quickly and clearly, Turbo Pascal is an ideal and convenient way to become acquainted with multi-tasking.

After you've read in the data strips, refer to Mr. Bergmann's article for operating instructions.

Reprinted with permission of Dr. Dobb's Journal.



1

2

3

Softstrip
COMPUTER SOFTWARE

EDITORIAL

Admirers of the late, lamented *Softalk PC* will experience a twinge of déjà vu on reading our table of contents this month: The Right to Assemble was a regular feature of that magazine, written by our own Ray Duncan. The writer in us regretted such an apt title going to dust, and the programmer in us wanted to see a regular column on assembly language in our pages, so the editor in us made some phone calls. Craig Stinson of *Softalk* and Ray graciously granted us use of the title. The column's contents will come from several authors and address several architectures; it will always include assembly-language code.

It's hard to let a good column title or idea atrophy. We got a call from Fred Davis at *A+* magazine asking of us what we had asked of Craig and Ray. As a result, Computer Calisthenics, the puzzle column launched here in *DDJ* by Michael Wiesenber, is now running in *A+*. Many of you will recall that the original complete title of *DDJ* was *Dr. Dobb's Journal of Computer Calisthenics and Orthodontia: Running Light Without Overbyte*. Now we just need to license the Orthodontia.

The real old-timers will be grumbling about our error in the last paragraph; the original title spoke of *Tiny BASIC*, not *Computers*, they will point out. True enough, and that's occasion for another note of avuncular pride: Gordon Brandy's February 1985 piece entitled "Tiny BASIC for the 68000" was just reprinted in *I/O*, a leading Japanese computer magazine. *I/O* has also reprinted Jim Hendrix's articles on Small-C. We're happy that *DDJ* can be involved in software development worldwide. In fact, since we believe that the community of serious programmers



is fundamentally one international community, we are beginning to make a greater effort to distribute the magazine internationally and to draw on authors from around the world.

This issue was too stuffed to fit in Chip Watch, Professional Programmer, or *DDJ On Line*, but here at least is the on-line report:

After a fitful start in January, the *DDJFORUM* is humming along nicely on CompuServe. We apologize to all who logged on January 1 only to find things in an unfinished state. We became fully operational on January 16 and since then have gradually added to our Data Libraries (DLs). You can now download all the code from issues 111, 112, and 113 (with the exception of Allen Holub's MS DOS shell and utilities) along with various selections from back issues.

We want the DLs to contain listings that our readers have asked for. If there is something from a back issue that is particularly useful to you, let us know. If demand is great enough, we will make it available.

Our message boards have also seen considerable activity. Columnists Ray Duncan and Allen Holub are sysops and log on regularly to answer queries and chat. As of the date this was written, no special conferences have been planned, but we suggest that you sign on and read the Conference Bulletin to find a more recent update.

Hope you will drop by soon.

A handwritten signature of Michael Swaine.

Michael Swaine

Dr. Dobb's Journal of Software Tools

Editorial

Editor-in-Chief Michael Swaine

Managing Editor Vince Leone

Editorial Assistant Sara Noah Ruddy

Technical Editor Allen Holub

Contributing Editors Ray Duncan

Allen Holub

Copy Editors Laura Kenney

Rhoda Simmons

Special Projects Editor Frank DeRose

Production

Production Manager Bob Wynne

Art Director Shelley Rae Doeden

Production Assistant Alida Hinton

Typesetter Jean Aring

Cover Artist Tim Gault

Circulation

Fulfillment and Stephanie Barber

Subscription Mgr. Maureen Kaminski

Book Marketing Mgr. Jane Sharninghouse

Circulation Assistant Kathleen Shay

Administration

Finance Manager Sandra Dunie

Business Manager Betty Trickett

Accounts Payable Supv. Mayda Lopez-Quintana

Accounts Payable Assts. Denise Giannini

Kathy Robinson

Billing Coordinator Laura Di Lazzaro

Accountant Marilyn Henry

Adm. Coordinator Kobi Morgan

Advertising

Advertising Director Shawn Horst (415) 366-3600

Advertising Sales

Walter Andruszewski (617) 868-1524

Lisa Boudreau (415) 366-3600

Michelle Beatty (317) 875-8093

Michael Wiener (415) 366-3600

Systems Manager Ron Copeland

Administrative Manager Anna Kittleson

Advertising Secretary Michelle A. Davi

M&T Publishing, Inc.

Chairman of the Board Otmar Weber

Director C.F. von Quadt

President and Publisher Laird Foshay

Dr. Dobb's Journal (USPS 3076900 is published monthly by M&T Publishing, Inc., 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600, Second class postage paid at Palo Alto and at additional entry points.

Address correction requested. Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, P.O. Box 27809, San Diego, CA 92128.

ISSN 0884-5395

Customer Service: For subscription problems call outside CA 800-321-3333; within CA 619-485-9623 or 566-6974. For book, back issue, or disk order problems call 415-424-1474.

Subscription Rates: \$29.97 per year within the United States. Foreign subscription rates: \$56.97 for airmail, \$46.97 for surface mail. Foreign subscriptions must be pre-paid in U.S. Dollars, drawn on a U.S. Bank.

Foreign Distributor: Worldwide Media Service, Inc., 386 Park Ave. South, New York, NY 10016, (212) 6686-1520 TELEX: 620430 (WUI)

Entire contents copyright © 1986 by M&T Publishing, Inc. unless otherwise noted on specific articles. All rights reserved.



People's Computer Company

Dr. Dobb's Journal is published by M&T Publishing, Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a non-profit, educational corporation.



The C for Microcomputers

PC-DOS, MS-DOS, CP/M-86, Macintosh, Amiga, Apple II, CP/M-80, Radio Shack, Commodore, XENIX, ROM, and Cross Development systems

MS-DOS, PC-DOS, CP/M-86, XENIX, 8086/80x86 ROM

Manx Aztec C86

"A compiler that has many strengths... quite valuable for serious work"

Computer Language review, February 1985

Great Code: Manx Aztec C86 generates fast executing compact code. The benchmark results below are from a study conducted by Manx. The Dhrystone benchmark (CACM 10/84 27:10 p1018) measures performance for a systems software instruction mix. The results are without register variables. With register variables, Manx, Microsoft, and Mark Williams run proportionately faster. Lattice and Computer Innovations show no improvement.

	Execution Time	Code Size	Compile/Link Time
Dhrystone Benchmark			
Manx Aztec C86 3.3	34 secs	5,760	93 secs
Microsoft C 3.0	34 secs	7,146	119 secs
Optimized C86 2.20J	53 secs	11,009	172 secs
Mark Williams 2.0	56 secs	12,980	113 secs
Lattice 2.14	89 secs	20,404	117 secs

Great Features: Manx Aztec C86 is bundled with a powerful array of well documented productivity tools, library routines and features.

Optimized C compiler Symbolic Debugger
 AS86 Macro Assembler LN86 Overlay Linker
 80186/80286 Support Librarian
 8087/80287 Sensing Lib Profiler
 Extensive UNIX Library DOS, Screen, & Graphics Lib
 Large Memory Model Intel Object Option
 Z (vi) Source Editor -c CP/M-86 Library -c
 ROM Support Package -c INTEL HEX Utility -c
 Library Source Code -c Mixed memory models -c
 MAKE, DIFF, and GREP -c Source Debugger -c
 One year of updates -c CP/M-86 Library -c

Manx offers two commercial development systems, Aztec C86-c and Aztec C86-d. Items marked -c are special features of the Aztec C86-c system.

Aztec C86-c Commercial System	\$499
Aztec C86-d Developer's System	\$299
Aztec C86-p Personal System	\$199
C-tree database (source)	\$399

All systems are upgradable by paying the difference in price plus \$10.

Third Party Software: There are a number of high quality support packages for Manx Aztec C86 for screen management, graphics, database management, and software development.

C-tree \$395	Greenleaf \$185
PHACT \$250	PC-lint \$98
HALO \$250	Amber Windows \$59
PRE-C \$395	Windows for C \$195
WindScreen \$149	FirTime \$295
SunScreen \$99	C Util Lib \$185
PANEL \$295	Plink-86 \$395

MACINTOSH, AMIGA, XENIX, CP/M-68K, 68k ROM

Manx Aztec C68k

"Library handling is very flexible... documentation is excellent... the shell a pleasure to work in... blows away the competition for pure compile speed... an excellent effort."

Computer Language review, April 1985

Aztec C68k is the most widely used commercial C compiler for the Macintosh. Its quality, performance, and completeness place Manx Aztec C68k in a position beyond comparison. It is available in several upgradable versions.

Optimized C	Creates Clickable Applications
Macro Assembler	Mouse Enhanced SHELL
Overlay Linker	Easy Access to Mac Toolbox
Resource Compiler	UNIX Library Functions
Debuggers	Terminal Emulator (Source)
Librarian	Clear Detailed Documentation
Source Editor	C-Stuff Library
MacRam Disk -c	UniTools (vi,make,diff,grep) -c
Library Source -c	One Year of Updates -c

Items marked -c are available only in the Manx Aztec C68-c system. Other features are in both the Aztec C86-d and Aztec C68-c systems.

Aztec C68k-c Commercial System	\$499
Aztec C68d-d Developer's System	\$299
Aztec C68k-p Personal System	\$199
C-tree database (source)	\$399
AMIGA, CP/M-68k, 68k UNIX	call

Apple II, Commodore, 65xx, 65C02 ROM

Manx Aztec C65

"The AZTEC C system is one of the finest software packages I have seen"

NIBBLE review, July 1984

A vast amount of business, consumer, and educational software is implemented in Manx Aztec C65. The quality and comprehensiveness of this system is competitive with 16 bit C systems. The system includes a full optimized C compiler, 6502 assembler, linkage editor, UNIX library, screen and graphics libraries, shell, and much more. The Apple II version runs under DOS 3.3, and ProDOS, Cross versions are available.

The Aztec C65-c/128 Commodore system runs under the C128 CP/M environment and generates programs for the C64, C128, and CP/M environments. Call for prices and availability of Apprentice, Personal and Developer versions for the Commodore 64 and 128 machines.

Aztec C65-c ProDOS & DOS 3.3	\$399
Aztec C65-d Apple DOS 3.3	\$199
Aztec C65-p Apple Personal system	\$99
Aztec C65-a for learning C	\$49
Aztec C65-c/128 C64, C128, CP/M	\$399

Distribution of Manx Aztec C

In the USA, Manx Software Systems is the sole and exclusive distributor of Aztec C. Any telephone or mail order sales other than through Manx are unauthorized.

Manx Cross Development Systems

Cross developed programs are edited, compiled, assembled, and linked on one machine (the HOST) and transferred to another machine (the TARGET) for execution. This method is useful where the target machine is slower or more limited than the HOST. Manx cross compilers are used heavily to develop software for business, consumer, scientific, industrial, research, and educational applications.

HOSTS: VAX UNIX (\$3000), PDP-11 UNIX (\$2000), MS-DOS (\$750), CP/M (\$750), MACINTOSH (\$750), CP/M-68k (\$750), XENIX (\$750).

TARGETS: MS-DOS, CP/M-86, Macintosh, CP/M-68k, CP/M-80, TRS-80 3 & 4, Apple II, Commodore C64, 8086/80x86 ROM, 68xxx ROM, 8080/8085/Z80 ROM, 65xx ROM.

The first TARGET is included in the price of the HOST system. Additional TARGETS are \$300 to \$500 (non VAX) or \$1000 (VAX).

Call Manx for information on cross development to the 68000, 65816, Amiga, C128, CP/M-68K, VRTX, and others.

CP/M, Radio Shack, 8080/8085/Z80 ROM

Manx Aztec CII

"I've had a lot of experience with different C compilers, but the Aztec C80 Compiler and Professional Development System is the best I've seen."

80-Micro, December, 1984, John B. Harrell III

Aztec C II-c (CP/M & ROM)	\$349
Aztec C II-d (CP/M)	\$199
C-tree database (source)	\$399
Aztec C80-c (TRS-80 3 & 4)	\$299
Aztec C80-d (TRS-80 3 & 4)	\$199

How To Become an Aztec C User

To become an Aztec C user call 1-800-221-0440 or call 1-800-832-9273 (800-TEC WARE). In NJ or outside the USA call 201-530-7997. Orders can also be telexed to 4995812.

Payment can be by check, COD, American Express, VISA, Master Card, or Net 30 to qualified customers.

Orders can also be mailed to Manx Software Systems, Box 55, Shrewsbury, NJ 07701.

How To Get More Information

To get more information on Manx Aztec C and related products, call 1-800-221-0440, or 201-530-7997, or write to Manx Software Systems.

30 Day Guarantee

Any Manx Aztec C development system can be returned within 30 days for a refund if it fails to meet your needs. The only restrictions are that the original purchase must be directly from Manx, shipped within the USA, and the package must be in resalable condition. Returned items must be received by Manx within 30 days. A small restocking fee may be required.

Discounts

There are special discounts available to professors, students, and consultants. A discount is also available on a "trade in" basis for users of competing systems. Call for information.

MANX

To order or for information call:

800-221-0440

LETTERS



Columns

Dear DDJ,

I noticed an apparent error in the listing of fgrep.c in *DDJ*, September 1985. The fifth line on page 63 should read *stoupper(str);* not *-stoupper(buffer);*. This would keep the *-y* flag from working unless a string file was used.

Michael J. Eager
481 Century Dr.
Campbell, CA 95008

Dear DDJ,

I got a red *Dr. Dobb's* T-shirt at the *Journal's* CP/M 82 (or was it 81?) booth. Now, alas, I need another piece of apparel to go with it: a black armband with the words "Bring Back Cortesi."

The old guard keeps changing at *Dr. Dobb's*. Nostalgia for the chip-smoking-hackers' glorious Portola Valley tabloid can't mask the fact that times are a changing out here for your readers too. No one cares anymore whether we can redirect PUN: to a solder-on UART. They care whether the shirt is white. Thank Peter Norton they don't care whether the collar is buttoned!

Losing Cortesi is another matter. He was current. He was relevant. He was beautifully articulate. And though I'd rather give up night skiing than Turbo Pascal, he was most often right.

Good luck, Dave, pursu-

ing whatever "interest having nothing to do with software" you're into.

Charles Landis
1803 39th Ave. East
Seattle, WA 98112

ProDOS

Dear DDJ,

I have been receiving *DDJ* in exchange for another publication, which has gone down the tubes. For the past six months I have glanced at your magazine. Undoubtedly, I would not have renewed my subscription. With the December 1985 issue, however, you have captured my attention. Imagine finding two articles pertaining to the Apple II world and, more particularly, to ProDOS, my favorite subject. If such events continue, I might become a disciple or even a contributor. What are *DDJ*'s plans for addicts like me? How about an is-

sue devoted to the 6502?

It would be out of character for me not to comment on Shawn Day's article, "Adding a Copy Command to ProDOS." I have several nits to pick and one severe deficiency to point out.

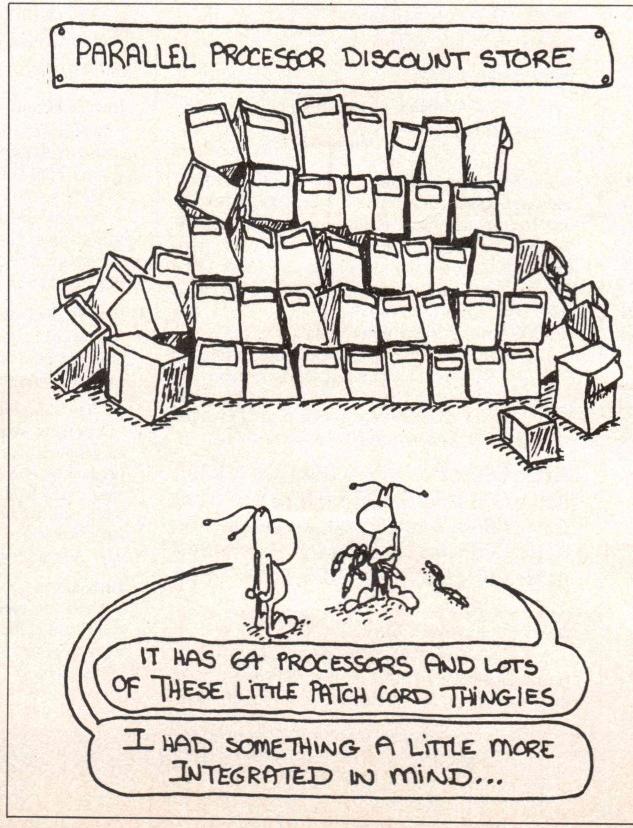
In first stage processing, a pointer is used for reading the text of the user's input from the primary text buffer. This could have been accomplished more directly by examining the input buffer. In keeping with ProDOS BASIC convention, spaces should have been ignored during command parsing. Finally, it is not necessary to implement prefix fetching and file creating in PBITS. If you doubt this assertion, assign values of 3 and 0 to PBITS and PBITS+1, respectively, and observe that program function does not change.

In second stage process-

ing, every direct call to the machine-language interface could be replaced by a call to GOSYSTEM, making program size smaller and program logic more consistent. This fault is minor, however, in comparison to the inability of the code (e.g., lines 361–378) to copy sparse files accurately. Before describing a solution to the problem, I shall briefly explain this generic file entity.

A sparse file contains discontinuous data (i.e., holes exist in the file). The number of data bytes that potentially can be read from the file exceeds the number of data bytes actually stored within the file. A random access text file (RATF) is the prototypical sparse file. Not all RATFs are sparse, but the potential exists. For example, consider a RATF with a record length of 512 bytes. One physical ProDOS block holds one file record. When you first open (i.e., create) the file, it has a length of 0 and is contained on one block. If next you write 10 bytes of data to Record 200, the file length becomes 102,410 bytes (i.e., $10 + [512 \times 200]$) but the file occupies only three blocks. If all of the prior records held data, the file would require 202 blocks. Thus, the file is sparse. This economy of space is made possible by index blocks which keep track of but do not allocate space for dataless records. Under ProDOS the most extreme example of this phenomenon is a file whose length is 16 megabytes and whose physical size is only 5 blocks. That's sparse!

When Shawn Day's code reproduces a sparse file, it assigns physical blocks to actual and potential data



SOURCE CODE
INCLUDED!

Mult-user, Multitasking Mainframe Performance from your PC, XT or AT . . .

... With WENDIN®
Operating Systems
and Software
Development Tools.

Operating System Toolbox™ only \$9900

Operating System Toolbox™ is a software construction set that enables you to build your own multitasking, multiuser operating system. All you need is your creativity and imagination to write a shell and link it with the Toolbox. Your Operating System will support your own commands, your WENDIN standard system services and your special user interface. Your Operating System will run most MS-DOS and PC-DOS programs without modification. Includes full source code written in Microsoft "C", organized into seven basic modules, all on disk. You also get object modules (in case you don't need to compile the whole Kernel), plus a fantastic in-depth, step-by-step instruction manual on how to build your personal operating system.

The best part of the news is the price — so low, you'll want to order today, just to keep up with the state of the art in operating systems.

ORDER HOTLINE
509/235-8088
Master Card & Visa

Foreign orders inquire about shipping.
Domestic orders add \$3.50/1st item, \$1.00 each additional item for shipping, handling and insurance.

Washington residents add 7.8% sales tax.



WENDIN®

BOX 266
CHENEY, WA 99004

The people who make quality
software tools affordable.

DEALER INQUIRIES WELCOME!

PC-DOS is a trademark of IBM.
MS is a trademark of Microsoft.
Unix is a trademark of AT&T.
VAX/VMS is a trademark of Digital Equipment Corporation.
Wendin and XTC are Registered Trademarks of Wendin, Inc.
PCVMS, PCUNIX, and Operating System Toolbox are Trademarks of Wendin, Inc.

PCUNIX™ only \$9900

PCUNIX™ is a true multitasking, multiuser operating system similar to the popular UNIX® operating system for AT&T, selling for thousands of dollars more. It includes nearly 50 utilities, designed to make your program development a snap.

PCUNIX™ runs most MS-DOS and PC-DOS programs, so that you'll never have to buy another set of development tools! You can use existing compilers, linkers and editors. Programs run under PCUNIX™ can use MS-DOS system calls plus over 70 enhanced system services found in our Operating System Toolbox™. You get 4 DSDD disks jam packed with the source code to the shell and all utilities hand crafted in Microsoft "C".

The source code to the Kernel comes with Operating System Toolbox™, sold separately.

PCVMS™ only \$9900

PCVMS™ is Wendin's version of the popular VAX/VMS operating system, which was developed by Digital Equipment Corporation for their line of VAX computers. PCVMS™ turns your IBM-PC, XT, AT or true compatible into a supercharged, multitasking, multiuser workstation that runs MS-DOS and PC-DOS programs. Programs run under PCVMS™ can use MS-DOS system calls plus over 70 enhanced system services found in our Operating System Toolbox™.

PCVMS™ comes with full source code to the PCVMS™ shell and its utilities, hand crafted in "C", and supports up to 3 users using additional serial terminals.

The source code to the Kernel comes with Operating System Toolbox™, sold separately.

XTC® only \$9900

XTC® is the world's first multitasking editor for the IBM-PC. It also runs on IBM-XT and IBM-AT computers, as well as true compatibles. Designed BY programmers FOR programmers, XTC is the ultimate editing tool for software developers using C, PASCAL, ASSEMBLY, BASIC, FORTRAN, and other languages.

Why is XTC® the ultimate tool for editing in YOUR development environment? Because it has powerful features like MULTITASKING MACROS, WINDOWS, TEXT BUFFERS, UNTON TIMES, MACRO PROGRAMMING CONTROL STRUCTURES and VARIABLES, as well as blinding speed that leaves other editors in the stone age.

XTC® comes with full source code written in Microsoft Pascal, on 3 DSDD disks.

LETTERS

(Continued from page 8)

blocks alike. Using my first example, his copy command would produce a file with 202 blocks instead of 3 blocks. That is not acceptable, especially since RATFs almost cry out to be copied to a RAM disk. What to do?

The principles involved in correcting the flaw are not difficult. When a file is opened, the buffer reserved for that file is 1024 bytes long. The lower half holds an image of the most recent data block being read or written. The upper half contains an image of the index block for the current 128K of file content. By reading the source file in one-block increments, checking whether data exists by testing the corresponding index bytes (e.g., zero means no data; non-zero indicates data), and writing to the target file only if data is found, the sparse file can be faithfully duplicated. File position must be monitored during the process so that data is written to the correct location within the target file. The SET_MARK and GET_MARK commands are ideal for this purpose. Admittedly, block-by-block transfer of file contents is slower than bulk movement, but the difference in speed is surprisingly small for files of 64K or less, and accuracy is the sine qua non for a file copy program.

If any reader would like an annotated listing of this process, I shall be happy to provide one for the price of a SASE.

Let's hear it for the 6502!

Sandy Mossberg

50 Talcott Rd.

Rye Brook, NY 10573

Bose-Nelson Sort

Dear DDJ,

Mr. Celko's article on the

Bose-Nelson sort (DDJ, September 1985) was very interesting and thought-provoking, particularly his comments on stable versus nonstable sorts. As an addendum to his article, I wish to offer the following remarks:

1. I was unable to get the recursive program (Listing One) to work. There appear to be at least two problems. First, in the procedure *bosesort*, the test *if(n>1)* is incorrect since *n* has not been declared. I suspect that this test should be *if(j>1)*. Second, the expression for *m* simplifies to *m=j/2*. This cannot be correct since the result is an endless sequence of recursive calls *bosesort(2, 2)*. Since the notation used in Listing One is different from the notation used in the algorithm statement (an unfortunate choice!) I

cannot deduce how *m* should be calculated.

2. As printed, Listing Two is also incorrect. In case 2 of procedure *bosesort*, the first call to *push3* should instead be a call to *push5*. Also, in case 3, *x* is defined twice. The second definition should be *y = stack/top-4*;

3. To compare the recursive and nonrecursive methods, I wrote the program shown in my Listing One, page 54, (borrowing heavily from Mr. Celko's Listing Two!). Note that my method of retrieving command line arguments is applicable to Small-C.

4. For some reason, the sequence of swaps generated by the recursive program is the reverse of the sequence produced by the nonrecursive version.

5. A brief comparison of the recursive version (my

Listing One) with the non-recursive version (his Listing Two) is shown in Table 2, below. Execution times have been corrected for loading time. Also, the timed programs did not print out the swap pairs.

6. The B-N algorithm generates 32,708 pairs for a list of 664 items. Therefore, if you must sort more than this many items, you must either use an unsigned integer for the variable count, or you must use a longer integer.

7. Table 1, below, compares the B-N sort with the theoretical minimum and with *nlog2(n)* for values of *n* up to 40. A few points should be noted. First, the B-N sort efficiency drops off fairly steadily as *n* increases. For *n=100*, the B-N algorithm generates almost three times the minimum number of swap pairs. However, the relationship is not smooth. For example, increasing from 95 to 96 items requires 7 additional swaps. Increasing to 97 items requires 64 additional swaps, and increasing to 98 items requires 32 additional swaps.

8. I have not had a chance to test the B-N strategy against the more popular sorting algorithms. Obviously, a B-N sort will take less time to sort a sorted or nearly sorted list (fewer swaps are required) than a random list. However, if we compare the number of swap pairs generated by the B-N algorithm with *n*log2(n)*, as shown in Table 1, it appears that for large values of *n* (say, 15 or higher) a Bose-Nelson sort may not be able to compete with a properly selected conventional sort.

Richard J. Wissbaum
15553 E. Wyoming Dr. - H
Aurora, CO 80012

<i>n</i>	Theory	B-N	<i>n*log2(n)</i>	<i>n</i>	Theory	B-N	<i>n*log2(n)</i>
1	0	0	0	21	66	118	93
2	1	1	2	22	70	125	99
3	3	3	5	23	75	133	105
4	5	5	8	24	80	138	111
5	7	9	12	25	84	154	117
6	10	12	16	26	89	163	123
7	13	16	20	27	94	173	129
8	16	19	24	28	98	179	135
9	19	27	29	29	103	191	141
10	22	32	34	30	108	198	148
11	26	38	39	31	113	206	154
12	29	42	44	32	118	211	160
13	33	50	49	33	123	243	167
14	37	55	54	34	128	260	173
15	41	61	59	35	133	278	180
16	45	65	64	36	139	288	187
17	49	81	70	37	144	308	193
18	53	90	76	38	149	319	200
19	57	100	81	39	154	331	207
20	62	106	87	40	160	338	213

Table 1: Comparison of Bose-Nelson Sort with Theoretical Minimum

	Recursive	Non-Recursive
.EXE file size	8633	14166
Execution Time (100 items)	1.3	2.6
Execution Time (1000 items)	54.6	110.9

Table 2

Mark Williams

Now the biggest name
in C compilers comes in a size
everybody can afford.

Let's C™

\$75

Introducing Mark Williams' \$75 C compiler. Want to explore C programming for the first time? Or just on your own time? Now you can do it in a big way without spending that way. With Let's C.

This is no little beginner's model. Let's C is a powerful programming tool, packed with all the essentials of the famous Mark Williams C Programming System. The one chosen by Intel, DEC, Wang and thousands of professional programmers. The one that wins the benchmarks and the reviewers' praise:

"(This compiler) has the most professional feel of any package we tested..."—BYTE
"Of all the compilers reviewed, (it) would be my first choice for product development."—David W. Smith, PC WORLD

And now for more big news. Get our revolutionary csd C Source

Debugger for just \$75, too. You can breeze through debugging at the C source level ignoring clunky assembler code.

Affordable, powerful, debuggable. Mark Williams Let's C is the big name C compiler at a price you can handle. Get your hands on it now.

- Mark Williams Let's C
- For the IBM-PC and MS-DOS
 - Fast compact code plus register variables
 - Full Kernighan & Ritchie C and extensions
 - Full UNIX™ compatibility and complete libraries
 - Small memory model
 - Many powerful utilities including linker, assembler, archiver, cc, one-step compiling, egrep, pr, tail, wc
 - MicroEMACS full screen editor with source
 - Supported by dozens of third party libraries
 - Upgradeable to C Programming System for large scale applications development

Let's C Benchmark Done on an IBM-PC/XT, no 8087. Program: Floating Point from BYTE, August, 1983.

Exec Time in Seconds

Let's C

134.20
347.45

ORDER NOW! 60-DAY MONEY BACK GUARANTEE!

Mark Williams Let's C

Please send me:

____ copies of Let's C and ____ copies of csd (C Source Debugger)
at \$75 each. (Ill. residents add 7% sales tax.)

Check Money Order Visa, MasterCard or
American Express

Name _____

Address _____

City _____ State _____ Zip _____

Card # _____ Exp. Date _____

Signature _____



Mark
Williams
Company

1430 West Wrightwood
Chicago, Illinois 60614

Shrink-wrapped Big Macs?

I read with great interest the editorial in the December 1985 *DDJ* concerning the recent spate of litigation in the software industry. I share the concern expressed regarding current trends in software protection and in particular with how many software publishers are claiming both copyright and trade secret protection for software packages that are widely distributed.

As an attorney specializing in patent, trademark, copyright, and other intellectual property law, I have always believed that patents should protect underlying ideas, and copyrights should protect the expression of ideas. The United States Supreme Court has clearly shown that a patentable invention exists even when the actual improvement exists solely in software. Consequently, if a programmer develops a software package that performs new and useful functions then the idea underlying the software may be patentable.

On the other hand, Copyright Statute 17 USC, (Section 101, et seq) specifically provides that copyright protection for an

Alfred A. Fressola

Alfred A. Fressola is a partner in the Law Offices of Mattern, Ware, Stoltz, & Fressola P.C., 34 Sherman Ct., P.O. Box 783, Fairfield, CT 06430. He is also a member of the Computer Law Section of The Connecticut Bar Association.

original work of authorship does not protect "any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work."

The National Commission on New Technological Uses of Copyrighted Works (CONTU), empowered by Congress to review copyright legislation, concluded that copyright protection should be afforded to computer programs and databases. It proposed changes in the copyright statute, including a provision stating that it would not be an infringement for a rightful possessor of a copy of a computer program either to use the program in conjunction with a machine or to make archival copies.

It is interesting that this section (Section 117) as enacted does not use the language "rightful possessor" but rather uses "owner of a copy." Although I have not been able to find any legislative history concerning the change, it is clear to me that the party or parties who caused this change were mindful that the owner of a copyrighted work may license the use of a copy of the copyrighted work as distinguished from selling a copy of the work. Therefore, if one could argue that what would normally be considered a sale of a copy was instead a license to use a copy then all bets would be off with respect to the aforementioned section and to Section 109(a) of the copyright law. Section 109(a) states that the owner of a particular copy is entitled, without the authority

of the copyright owner, to sell or otherwise dispose of the possession of that copy.

Although a license can certainly be entered into between two parties, it requires more than the opening of a package to show acceptance of terms of the license; it is also outside the intention of CONTU that trade secrets be preserved by licensing copies wherein hundreds or thousands of licenses are made in such a manner. The software subcommittee report from CONTU states that it is self-evident that trade secrecy cannot be relied upon if a program is widely marketed or otherwise broadly distributed.

Some software publishers have been trying to have their cake and eat it too. They claim to have copyright protection and are only granting licenses to use copies of the copyrighted work, such licenses being accepted by the user opening a package. By their reasoning, virtually any item, not just software, could be packaged in a shrink-wrap form containing a "license" limiting all warranties, liabilities, and even ownership of the item! Certainly that was not the intention of CONTU and was an oversight by Congress when the phrase "rightful possessor" was changed to "owner of a copy."

I am in the process of investigating this matter further and may recommend that legislation be introduced to amend Section 117 of the Copyright Statute. A change in the statute is the proper avenue to clarify the right of the purchaser of a computer program to make archival copies and to analyze the

program. If the publisher wishes to protect underlying inventive ideas it should seek such protection by way of the patent statutes, which provide protection for a 17-year term in exchange for full disclosure of the invention. I would appreciate comments and thoughts because any change should be supported by all interested parties seeking to have fair protection afforded the authors of computer programs.

DDJ

NEW RELEASE

Ecosoft's Eco-C88 Rel. 3.0 C Compiler



\$5995

Release 3.0 has new features at an unbelievably low price. ECO-C88 now has:

- Prototyping (the new type-checking enhancement)
- enum and void data types
- structure passing and assignment
- All operators and data types (except bit fields)
- A standard library with more than 200 functions (many of which are System V compatible for greater code portability)
- cc and mini-make that all but automates the compile process
- 8087 support (we sense the 8087 at runtime - no dual libraries)
- ASM or OBJ output for use with MSDOS linker
- Tiered error messages - enable-disable lint-like error checking
- Fast compiles and executing code
- Expanded user's manual
- Enhanced CED program editor (limited time offer)

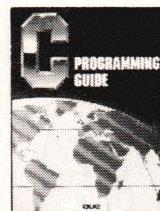
We also offer the following support products for Eco-C88.

CED Program Editor

CED now supports on-line function help.

If you've forgotten how to use a standard library function, just type in the name of the function and CED gives you a brief summary, including function arguments. CED is a full screen editor with auto-flagging of source code errors, multiple windows, macros, and is fully configurable to suit your needs. You can edit, compile, link, and execute DOS commands from within the editor. Perfect for use with Eco-C88. For IBM PC, AT and look alikes.

\$2995



C Programming Guide \$20

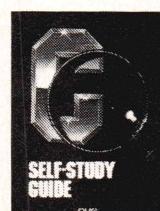
After reading the 1st edition, Jerry Pournelle (**BYTE Magazine**) said: "I recommend this book... Read it before trying to tackle Kernighan and Ritchie." The second edition expands this best seller and walks you through the C language in an easy-to-understand manner. Many of the error messages include references to this book making it a perfect companion to Eco-C88 for those just starting out with C.

C Source for Standard Library

Contains all of the source code for the library functions that are distributed with Eco-C88, excluding the transcendental and functions written in assembler.

\$10

(\$20 if not with order)



C Self-Study Guide

\$17

(Purdum, Que Corp.). Designed for those learning C on their own. The book is filled with questions-answers designed to illustrate many of the tips, traps, and techniques of the C language. Although written to complement the Guide, it may be used with any introductory text on C.

Developer's Library

Contains the source code for all library functions, including the transcendental and those written in assembler. Perfect for the developer that wish to write their own custom functions or learn how we implemented the Eco-C88 library.

\$25

(\$50 if not with order)



C Programmer's Library

\$20

(Purdum, Leslie, Stegemoller, Que Corp.). This best seller is an intermediate text designed to teach you how to write library functions in a generalized fashion. The book covers many advanced C topics and contains many useful additions to your library including a complete ISAM file handler.

ISAM Library

Contains the code from the C Programmer's Library in relocatable format (i.e., OBJ) including the delete code for the ISAM file handler.

\$15

(\$30 if not with order)

1-800-952-0472 (for orders)
or
1-317-255-6476 (tech. info.)

Eco-C88 C compiler requires an IBM PC, XT, or AT (or compatible) with 256K of memory, 2 disk drives and MSDOS 2.1 or later. Call today:



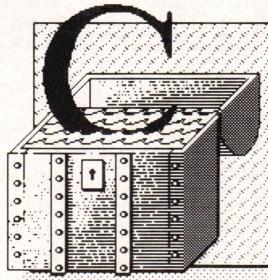
Ecosoft, Inc.

6413 N. College Ave. • Indianapolis, IN 46220



TRADEMARKS: ECO-C88, ECOSOFT

The Last of the Shell Support Routines



The code this month is the last of the shell support routines. A few subroutines are useful enough as stand-alone library routines to mention here, but for the most part I'll let the code speak for itself.

Ssort() (Listing Ten, page 70) is another general-purpose sort routine modeled after the Unix *qsort()* routine (See C Chest, April 1985). The April column implemented the routine as a Quicksort, whereas this version uses an adaptation of the Shell Sort in K & R. For many applications (especially those with only a small amount of data to sort or those where the data has already been sorted) a Shell Sort is a better way to go than a Quicksort. It uses much less code, and for small *N*, it's about as efficient as Quicksort. Like *qsort()*, *ssort()* is passed four arguments—the base address of the array to be sorted (*base*), the number of elements in the array (*nel*), the width of one element in bytes (*width*), and a pointer to a comparison function that is passed pointers to two elements in the array and should otherwise work like *strcmp()* does. You can find more details on how to use this routine and on how a shell sort works in the April C Chest.

Reargv() (Listing Nine, page 68) is not so much part of the shell itself as a routine for other programs that are being run under the shell. It's passed pointers to *argv* and *argc*, and it replaces these with another version created from the *CMDLINE* environment variable generated by the shell when a command is executed. The shell's complete (up to 2,048-byte) command line is passed to a program using this mechanism, because you

by Allen Holub

can pass only 127 characters via DOS. *Reargv()* should be called before *argv* or *argc* are used in *main()*. Note

that the program name is available as *argv[0]* if you use *reargv()*. This is not the case in DOS Versions 1 or 2.

If possible, you should incorporate the code from *reargv()* directly into your root module (where it belongs). It's provided here as a separate function because the Microsoft compiler doesn't give you the root module sources, so you have no choice but to call an independent subroutine.

Dir() (Listings Seven and Eight, pages 56 and 57) is a modified version of the directory routines that were part of the *ls* directory utility given in the July C Chest. *Dir* is passed a pointer to an initialized structure (called a *DIRECTORY*) containing various commands. It gets a directory and puts it into an *argv*-like array of pointers to strings, part of this structure. *Mk_dir()* creates a *DIRECTORY* structure. You set various flags in the structure to tell *dir()* what to do and then call it with a file specification and a pointer to the initialized *DIRECTORY* as its arguments. When the subroutine returns, various fields of the *DIRECTORY* will have been updated to hold the proper file or subdirectory names as well as other information such as file sizes and a volume label.

The returned directory is sorted for you (using *ssort()*). The routine *del_dir()* deletes a *DIRECTORY* structure. *Dir.h* (Listing Seven) has details about how to initialize the *DIRECTORY* and how *dir()* modifies it. If you want to see an example of how *dir()* is used, look in the shell on lines 478–566 of Listing One (January 1986). Refer to the July C Chest for details about getting directories from DOS in general.

Dir works in slightly different

ways from the *ls* described in July. The major difference is the total file size. If you specify a long format listing, the file size in bytes is still listed as part of the *dirv* entry. This size is the number of actual characters in the file. The total byte count, however, reflects the number of bytes actually occupied by the file on the disk. In DOS the minimum number of bytes allocated to a file is a "cluster," which is a contiguous group of disk sectors. On the PC/AT hard disk, a cluster requires 2K (four 512-byte sectors); 2K are required to store a file, even if that file contains only one character. The totals returned by *dir()* will reflect the file sizes, rounded up to the cluster size for your particular disk. (It varies from disk type to disk type—1K clusters are used on a 354K double-sided floppy.)

Lvalues

A few months back I mentioned a problem I was having with the Lattice C compiler. In particular, I wanted to be able to say:

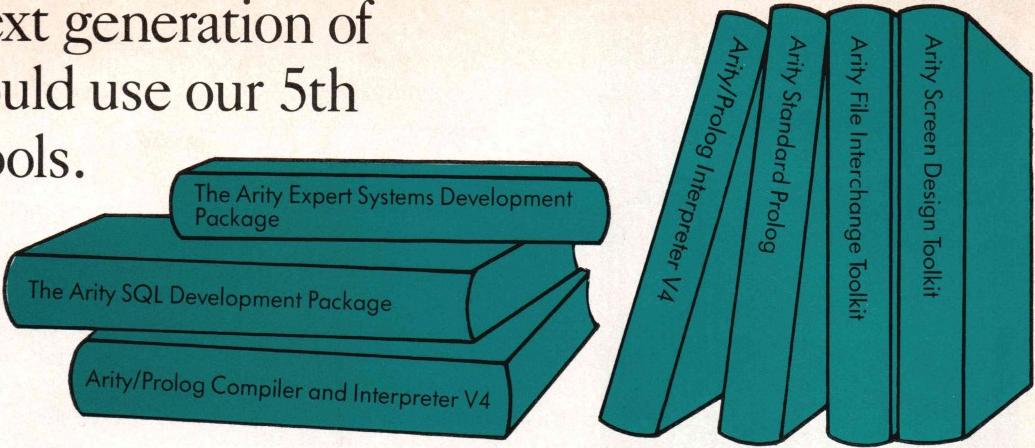
```
long    x;
int    *p;
x = *(long *p)++;
```

in order to be able to fetch a long-size object pointed to by *p* and then increment *p* as if it were a pointer to *long*. Lattice kicks this code out with an "lvalue required" error message. The code:

```
long    x;
char   *p;
x = *(long *p);
p += sizeof(long);
```

is accepted by Lattice. I think the first expression ought to compile, and in support of my contention, several compilers do indeed accept the statement and generate the correct code:

Why your next generation of products should use our 5th generation tools.



Arity's integrated family of programming tools allows you to combine software written in Arity/Prolog, the best of the fifth generation languages, with Arity SQL, the best of the fourth generation languages, and with conventional third generation languages such as C or assembly language to build your smarter application.

You can use Arity/Prolog to build expert systems using the Arity Expert Systems Development Package. Or to build natural language frontends. Or to build intelligent information management systems. Arity/Prolog lets you build advanced technology into your vertical applications package.

And more...

That's not the whole story. Arity's products are all designed to be fast, powerful, serious. Each of our products contains unexpected bonuses. Such as a one gigabyte virtual database integrated into Arity/Prolog. The most powerful of its kind on a PC.

Quality first. Then price.

In order to be the best, we had to prove it to our customers. Our tradition of quality software design is reflected in every product we sell. Quality first. Then price. And we always provide the best in customer support.

Our products are not copy protected. We do not charge royalties. We offer generous educational and quantity discounts. And we have a 30 day money back guarantee.

Try us to know that we keep our promise on commitment to quality and reliability. Try us by using our electronic bulletin board at 617-369-5622 or call us by telephone—you can reach us at 617-371-2422.

Or fill in this coupon. Whether you order today or not, let us send you full descriptions of our integrated family of Arity products.

arity

We design and distribute high quality, serious application software for the IBM PC, XT, AT and all MS-DOS compatibles.

Circle no. 121 on reader service card.

Please complete this form to place your order and/or request detailed information.

Quantity _____ Info only _____

Arity/Prolog Compiler and Interpreter V4	\$795.00
Arity/Prolog Interpreter V4	\$350.00
Arity Standard Prolog	\$ 95.00
Arity SQL Development Package	\$295.00
Arity Expert System Development Package	\$295.00
Arity Screen Design Toolkit	\$ 49.95
Arity File Interchange Toolkit	\$ 49.95
TOTAL AMOUNT (MA residents add 5% sales tax) (These prices include shipping to all U.S. cities)	\$ _____

Quantity _____ Info only _____

NAME _____

SHIPPING ADDRESS _____

CITY/STATE/ZIP _____

TELEPHONE _____

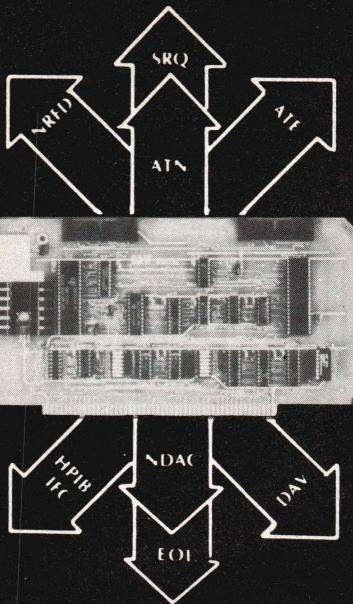
Payment: Check PO AMEX VISA MC

Card # _____ Exp. date _____

Signature _____

ARITY CORPORATION • 358 BAKER AVENUE • CONCORD, MA 01742

arity



THE 488+3

IEEE 488 TO S-100

INTERFACE

S-100 ↔ 488

- Controls IEEE 488 (HP1B) Instruments with an S-100 computer
- Acts as controller or device
- Basic and assembly language drivers supplied
- Meets IEEE 696 specification
- Industrial quality burned in and tested up to 125K bytes/sec under software control 3 parallel ports (8255-5)
- \$375



D&W DIGITAL, INC.
20655 Hathaway Avenue
Hayward, California 94541
(415) 887-5711

Circle no. 265 on reader service card.

C CHEST

(Continued from page 14)

Microsoft C (Version 3.0), Aztec C (Version 3.20d), and Whitesmith's C for the 68000 are three of these. Jim Howell writes that the Introl C compiler for the 6809 also generates correct code. Jim also mentions that the C-systems C compiler outputs garbage but doesn't report an error.

Several people have written about this problem, most of them advancing the argument that the Lattice C compiler is behaving rationally. In order to understand the problem better, we should start by defining *lvalue*. K & R say: "An *object* is a manipulatable region of storage; an *lvalue* is an expression referring to an object. An obvious example of an *lvalue* is an identifier. There are operators that yield *lvalues*: for example if E is an expression of pointer type, then *E is an *lvalue* expression referring to the object to which E points. The name '*lvalue*' comes from the assignment expression E1=E2 in which the left operand E1 must be an *lvalue* expression."¹

So, an *lvalue* is the place where the results of an expression evaluation are stored. As J. S. Williams writes, however, an "*lvalue* is not just an expression on the left-hand side of an assignment statement. A better interpretation is that the *lvalue* of an expression is the symbolic location of the value of the expression." That is, it's an address into which the result of the expression evaluation will be put. An *lvalue* has to be a real place—a variable that's been declared somewhere or the place pointed to by a pointer.

Several people mention a complementary concept to an *lvalue*—an *rvalue*. An *rvalue* represents the value of an *lvalue*. That is, if an *lvalue* is the address into which we'll put the result of an expression, an *rvalue* is the result itself—the number that will be put into the *lvalue*. Consulting the grammar in Appendix A of K & R (pages 214–219), there's no nonterminal symbol called an *rvalue*. That is, the concept of *rvalue* can be used to describe several nonterminal symbols in the grammar, whereas an *lvalue* is actually a nonterminal. Thus, talking about *rvalues* might help us to understand the problem conceptually, but the *rvalue* is not bolted

into the language as is the *lvalue*.

Nevertheless, an *rvalue* is a useful concept. The creation of an *rvalue* may involve memory allocation over which you have no control. Compilers sometimes put intermediate results of an expression evaluation into scratch variables (or registers) that they allocate transparently to your program. Thus, an *rvalue* may be associated with an "anonymous temporary" variable that you cannot access directly.

Wayne Throop writes, "A cast is defined in K & R (page 42) like this: 'In the construction '(type-name) expression,' the expression is converted to the named type by the conversion rules above. The precise meaning of a cast is in fact as if the expression were assigned to a variable of the specified type, which is then used in place of the whole construction.' Now, this paragraph can possibly be interpreted to mean that a cast must be an *lvalue*. Most compilers, however, do not so implement it. The key is in the 'converted to the named type' phrase. A cast (potentially) changes the bits of the cast expression. Anonymous temporaries are normally not treated as *lvalues* because things assigned to them can never be retrieved later."

In other words, a cast generates an *rvalue*, and as a result of evaluating the cast, the number might be copied into a scratch variable in order to ease any necessary arithmetic manipulation associated with the type conversion (such as sign extension, truncation of high-order bits, and floating point to integer conversion). So, in

`x = *(long *)p)++;`

the `(long *)p` may cause `p` to be copied to a temporary variable as part of the cast operation, and the `++` would increment the temporary variable, not `p` itself.

The point about temporary variables is well taken. Nevertheless, just because it's easiest for the compiler to always put a cast variable into an anonymous temporary, this behavior is not required by the language. More to the point, just because a variable might be copied somewhere else, there's no reason why the compiler can't copy it back when it has fin-

ished with it. In any event, *p* doesn't have to be copied anywhere in the problem we're discussing. We don't need to modify it; *long* pointers and *int* pointers are the same physically. The difference lies in how the object pointed to is used, not in any dissimilarities between the pointers themselves. By the same token, if *p* isn't copied to a temporary variable, there's no reason why you can't increment it, and several compilers will indeed let you increment a cast pointer.

To my mind, the human being shouldn't have to do something (or not do something) for the convenience of the compiler; rather, the compiler should do its best to behave rationally, provided that the input is rational. A compiler that makes intelligent decisions about when to copy and when not to copy variables is a better compiler—there's no point in copying objects that don't need to be copied. Unfortunately, many compilers—and Lattice C seems to be one of these—don't operate intelligently.

As a footnote, a more technically correct argument against incrementing a cast variable is that there's no obvious way to parse such an expression using the formal grammar in the back of K & R. Because several compilers accept the construct, it's obviously possible to write a grammar in which incrementing a cast variable is acceptable. More to the point, the grammar in K & R isn't used "as is" in any C compiler that I know of. Everybody (including K & R) has modified it a little to make it more realistic when they write a real compiler. The grammar in my copy of the draft proposed ANSI standard bears little relation to the one in K & R, so until the ANSI standard is formalized, I don't think the grammatical argument is valid either.

There is one argument that I do think is valid. As I've learned the hard way, *x = *((long *)p)++* is not portable, at least it's not portable to Lattice. On the other hand, *x = *(long *)p; p += sizeof(long)* is acceptable to all compilers. This second method is therefore preferable for portability reasons.

Problems with The Microsoft C Compiler

A few months ago Microsoft sent me

a review copy of the new version (3.0) of its C compiler. I've been using it since then (to develop the shell I've been talking about since January, among other things).

I really want to like this compiler. It generates compact code, and its implementation of the language is one of the best I've found (it supports enumerated types and strong type checking à la the proposed ANSI standard). Moreover, its library documentation is organized quite nicely with all the functions in alphabetical order instead of grouped by function (a prac-

tice I never could understand). You can find the documentation for any routine in the library quickly just by looking it up. There's no need to consult a cross-reference, then an index, and then look it up. Microsoft has tried to minimize the number of functions described by a single entry (very little of the one-page-for-15-similar-functions business), and every entry has a See Also section and an Examples section. I wish every compiler's library documentation was organized as nicely as Microsoft's.

C PROGRAMMERS, CALL A POWER PLAY WITH...



...db_VISTA DATABASE MANAGEMENT SYSTEM FOR C.

db_VISTA is a full-featured programmer's DBMS. It handles your data powerfully, yet economically without the frills that make end-user DBMS's bulky, slow and expensive to license. Use only the features you need for maximum efficiency with minimum code and effort.

Powerful lineup of features. B-tree indexing, multiple key records, transaction processing, interactive database access utility, and file transfer utilities for dbASE, R:base and ASCII files. You even get 90 days applications development support free of charge.

Define your playbook up front. As a network model DBMS, **db_VISTA** is suited to applications development. A premium is placed on efficient use of disk storage, reduced data redundancy and fast access times allowing you to cross the goal line first.

It's your game plan. The database structure is specified by you in **db_VISTA**'s data definition language (DDL). The DDL processor compiles the specification into tables (data dictionary) used by **db_VISTA**'s library functions, which are called by your C program to manipulate and access the database.

db_VISTA's written in C so you can understand the signals. Source code is optional. No sweat. No royalties, either.

All this delivered for less than the price of season tickets.

Single user without source	\$195
Single user with source	\$495
Multi-user without source	\$495
Multi-user with source	\$990

Available for most popular C compilers under **MS-DOS, XENIX, plus most UNIX systems.**

Go for the power play and order **db_VISTA** now. Call (206) 747-5570 or

1-800-843-3313

at the tone touch 700-992. 30 day money-back guarantee.

RAIMA
CORPORATION

12201 S.E. Tenth Street
Bellevue, WA 98005 USA
Telex: 9103330300

MACINTOSH VERSION
AVAILABLE NOW!

Circle no. 206 on reader service card.

C CHEST

(Continued from page 17)

The library is one of the more complete that I've seen, too, especially in the operating system support area. For example, it has both a *getenv* and a *putenv* routine so you can not only retrieve a variable from the current environment but you can add a variable to it as well. *Putenv* lets you pass variables to a spawned process pretty easily, without having to go through a temporary file. The compiler also comes with a Unix-like ver-

sion of the cc driver program. (It can compile and link your source code in one step.)

Unfortunately, the compiler itself has several serious problems. Like most compilers, the Microsoft compiler is actually several programs that are invoked in sequence by a controlling driver program. The various programs communicate with each other via temporary files and the command line. The first things I tried to compile were the hyphenation routines from October's C Chest column, and the compiler wouldn't

do it. Pass 3 of the compiler kicked out an incomprehensible error message that referred to a line number in a temporary file that was erased by the driver program (or so I assume—the file was not to be found on the disk). So, I thought to myself, I'll just run the passes by hand instead of letting the supplied driver program do it for me. No good. The individual passes of the compiler are not documented—anywhere. You can't run them yourself. More to the point, were I a naive user the situation would be more incomprehensible because the documentation doesn't explain that compilation involves the invocation of programs that you know nothing about. Anyway, I eventually fixed the problem by stripping the large arrays of initialized data out of the offending file and putting them into a second file. The compiler's problem seems to have something to do with symbol table space overflow, but who knows?

Another related problem—the compiler lets you use several environment variables to pass it information. The INCLUDE environment, for example, can hold a list of places to look for #included files. It seems, however, that the driver program knows about these environments, but the other programs that make up the compiler do not. In particular, the driver program passes the contents of the INCLUDE environment to the subprograms using the DOS command line. The DOS command line is limited to 127 bytes, so if the INCLUDE environment has too many characters in it, the command line can be truncated and the invoked subprogram will spit out an incomprehensible error message. Again, this behavior is not documented. You are not told to limit the length of the INCLUDE environment.

The above problems are characteristic of a general nonchalance toward error processing on the part of Microsoft. The compiler's normal error processing, even when it's working properly, is among the worst I've ever seen. Its error messages are for the most part uninformative and error recovery is nonexistent. For example, the message

"Syntax error '{'"

was generated by a missing semico-

ATRON BUGBUSTERS GREASE BORLAND LIGHTNING

"If we were starting a software company again, from scratch, Atron's AT PROBE™ would be among my very first investments. Without Atron's hardware-assisted, software debugging technology, the flash of Turbo Lightning™ would be a light-year away."

Philippe Kahn, President, Borland

HOW BORLAND DOES SO MUCH, SO WELL, SO FAST

We asked Borland International president Philippe Kahn to share his secrets for rapidly taking a good idea and turning it into rock-solid reality. How does the Borland team do so much, so well, so fast?

He begins, "I remember when Atron used the June 24, 1985 *Wall Street Journal* chart of top-selling software in an ad." [Note: At that time, seven of the top ten software packages were created by Atron customers; it's now nine out of ten.] "Side-Kick was number four, and I let Atron quote me in saying that there wouldn't have been a Side-Kick without Atron's hardware-assisted debuggers.

"You might say lightning has literally struck again. Turbo Lightning made number four on

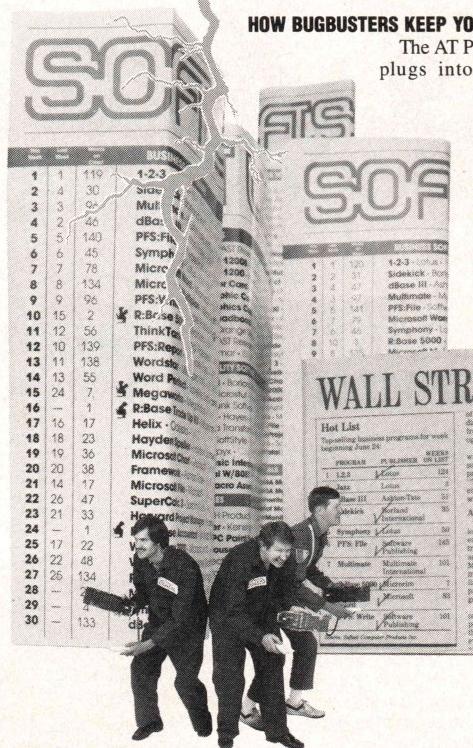
SoftSel's Hotlist within weeks of its introduction! And again, I say we couldn't have done it without Atron debugging technology.

"Cleverly written code is, by definition tight, recursive, and terribly complex," he continues. "Without the ability to externally track the execution of this code, competent debugging becomes very nearly impossible."

Concludes Philippe, "And after Turbo Lightning was solid and reliable, Atron tuning software turned our Probes into performance analyzers. How do you think we greased our lightning?"

Philippe, along with a couple million or so of your satisfied customers, we say congratulations on yet another best-selling product. We can't wait to see what awesomely useful technology will come shooting out of Borland International next.

Copyright © 1985 by Atron Corp. PC PROBE™ and AT PROBE™ Atron Sidekick™ and Turbo Lightning™ Borland International, Inc., Adv. by TRBA, 408/258-2708.



HOW BUGBUSTERS KEEP YOU FROM GETTING SLIMED

The AT PROBE is a circuit board that plugs into your PC/AT. It has an umbilical which plugs into the 80286 socket and monitors all 80286 activity.

Since AT PROBE can trace program execution in real time, and display the last 2048 memory cycles in symbolic or source-code form, you can easily answer the questions: "How did I get here?" and "What are those silly interrupts doing?"

It can solve *spooky* debugging problems. Like finding where your program overwrites memory or I/O—impossible with software debuggers.

You can even do source-level debugging in your favorite language, like C, Pascal or assembler. And after your application is debugged, the AT PROBE's performance measurement software can isolate performance bottlenecks.

Finally, the AT PROBE has its own 1-MByte of memory. Hidden and write-protected. How else could you develop that really large program, where the symbol table would otherwise take up most of memory.

LOOK AT IT THIS WAY.

History shows that non-Atron customers don't stand a very good chance of making the Top Ten list. Lightning *really does* have a way of striking twice!

The PC PROBE™ is \$1595 and the AT PROBE is \$2495. So call Atron today. You can be busting some really scary bugs tomorrow. And maybe, just like Borland, you can also bust some records.

Atron
THE DEBUGGER COMPANY

20665 Fourth Street • Saratoga, CA 95070 • 408/741-5900

Circle no. 216 on reader service card.

New from Logitech.

MODULA-2/86 VERSION 2.0

Professional
Modula-2
for \$89.

Now the same powerful tools Micropro used to develop its latest word processing system is available to you at a new \$89.00 price.

Systems to Fit Your Needs.

Base Language System

- Compiler and Linker
- Module Library

Base Language System/8087

- Inline 8087 code.

Base Language System/512K

- Full 8087 support.
- Uses RAM to increase speed by 40 to 50 percent.
- 80186 and 80286 support.

Run-Time Debugger

- Monitors the execution of a program with user-defined breakpoints or by stepping through the program.
- Symbolically displays the source code, data, procedure call chain, and raw memory.

MODULA-2 Editor

- Fast on-line Modula-2 syntax check.
- Can run compiler and linker from the editor.
- User definable templates for Modula-2 syntax constructs.

Utilities Package

- Decoders: Disassemble link and load files.
- Version: Administrate different versions of one program.
- Post-Mortem Debugger: Debugs a program after abnormal termination.
- Cross Reference: Produces a cross-reference listing of a Modula-2 program.

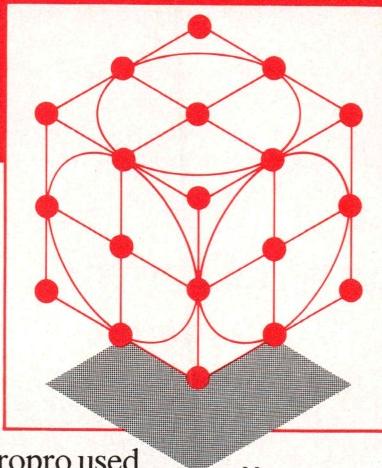
Sources

- Sources to customize your system.
- Run-Time System sources.
- Some library module sources.

Not Copy Protected

INTRODUCTORY OFFER

Through the end of March you get the new MODULA-2 Editor for free with any purchase of the Base Language System.



Building Blocks for Tomorrow's Technology

Universities are switching to LOGITECH MODULA-2. Innovative programmers now develop applications and products with LOGITECH MODULA-2. The most productive teams at major companies depend on LOGITECH MODULA-2.

Now you can create your professional software development system using the proven technical sophistication of LOGITECH MODULA-2/86.

To place an order call our special toll free number:

800-231-7717

In California:

800-552-8885

YES, I want to create my professional software development system. Please send me the following building blocks:

BLS \$89 BLS/8087 \$129 BLS/512K \$189
 RTD \$69* EDITOR \$59*

UTILITIES \$49* SOURCES \$179*

*\$10 less with the purchase of any Base Language System.
Please add \$5 for shipping and handling.

VISA MASTERCARD CHECK ENCLOSED

CARD NUMBER

EXPIRATION DATE

SIGNATURE

NAME _____

ADDRESS _____

CITY _____

STATE _____ ZIP _____ PHONE (____) _____



LOGITECH

LOGITECH, Inc.
805 Veterans Blvd., Redwood City, CA 94063, USA

Telephone: (415) 365-9852

LOGITECH SA
Box 32, CH-1143 Apples, Switzerland
Telephone: 41 (21) 774545

ion several lines above the one that was flagged in the error message. The compiler is showing you the symbol following the semicolon, the one that confused it. Because a missing semicolon is perhaps the most common error in a C program, you'd think that Microsoft would at least handle this one carefully—but no. Semicolons weren't mentioned in any of the hundred spurious error messages that followed. (I'm not ex-

aggerating the number of messages either, there really were a hundred of the things.) And then, to add insult to injury, the compiler aborted, telling me that there were too many errors to continue processing even though the missing semicolon was the only real error in the file.

This sort of behavior is repeated with disgusting regularity. A missing semicolon, colon, parenthesis, or curly bracket is a disaster. Misspell a keyword, and all is lost. Spurious error messages are a serious problem in themselves, and they can't be

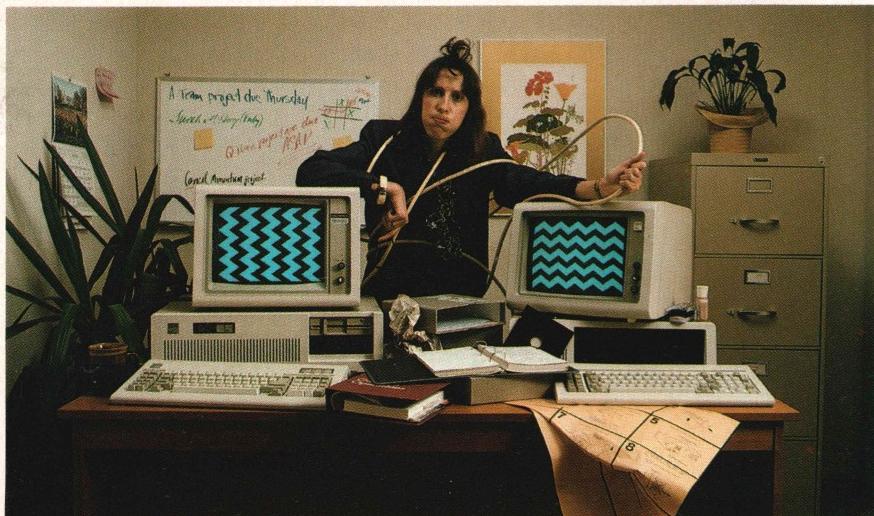
eliminated entirely, but a well-designed compiler ought to be able to recover from an error within a few statements and then go on processing. At the very least, it could skip to the next subroutine declaration and then start processing from there (you can do that with a regular expression). As far as I can tell, though, the Microsoft compiler makes no attempt to recover.

These goings-on dramatically slow a program's development time. Instead of compiling, fixing five or ten errors, and then recompiling, I have to compile, fix only one error, compile again, fix one error, and so forth. I may have to recompile 10 or 15 times to get all the errors fixed, a very time-consuming process to say the least. I don't care how fast a compiler is, if I have to recompile 15 times to find 15 errors, I'll never finish my program.

Another real problem here is the error message not telling me what the error was. If I were not already a reasonably proficient C programmer, I'd never have found some of my errors (which were sometimes several lines above the one flagged by the error message). If you don't already know C very well, the Microsoft compiler is a waste of money. You'll get so disgusted with its behavior, you'll never learn the language well enough to debug any but the simplest program.

Another problem is that the documentation doesn't mention various important differences between the way a Microsoft library routine works and the way that the equivalent Unix routine works. The most serious of these problems that I've found so far is in the *spawn* function. It turns out that files opened by a parent process are closed when the child process terminates! (And I don't use the ! lightly.) It seems OK for a child to inherit copies of the parent's file descriptors (which is documented), but the child shouldn't be able to modify them, much less close them. Even worse, it is an *exit()* call in the child that does the closing, not an explicit call to *close()*. In other words, there is no way for a file to stay open after a child is spawned. Maybe this isn't a problem with the *spawn* function at all, maybe it's a bug in MS DOS. In any event, bug or not, this aber-

If you can't share files on your network, you're using the wrong file manager.



Be connected. Btrieve.®

Networks can solve problems. But running a single-user file manager can create new ones: Lost updates. Garbled data. Trashed files.

Btrieve® /N offers safe multi-user file management that protects your data when sharing files. And eliminates the need to rewrite your application for LANs. Btrieve/N sets the file management standard for the industry's most popular networks, as well as XENIX and other multi-user systems.

Fast. Btrieve/N is fast, too. It's written in assembly language especially for the IBM PC. And based on b-tree file indexing, for access speed that won't degrade as your database grows.

Automatic file recovery. Btrieve/N provides automatic file recovery after a system crash. Your Btrieve data al-

ways comes back intact.

Fully-relational data management. SoftCraft's entire family of products gives you a complete, fully-relational database management system. Rtrieve™/N adds report writing capabilities. Xtrieve™/N speeds users through database queries with interactive menus.

For professional programmers.

Btrieve/N is the fast, reliable answer for all your application development in BASIC, Pascal, COBOL, C, FORTRAN and APL. With Btrieve/N, you can develop better network applications. And solve problems, not create new ones.

SC **SoftCraft Inc.**

P.O. Box 9802 #917 Austin, Texas 78766
(512) 346-8380 Telex 358 200

Suggested retail prices: Btrieve, \$245; Btrieve/N, \$595; Xtrieve, \$195; Xtrieve/N, \$395; Rtrieve, \$85; Rtrieve/N, \$175. Requires PC-DOS or MS-DOS 1.X, 2.X, or 3.X. NO ROYALTIES.

Circle no. 113 on reader service card.

rant behavior should be documented. There isn't a word about it in the manual, and it took me half a day to track it down.

There are a few other inconsistencies (such as *strncpy* pads out the string with nulls to the maximum count—why?), but I won't go into all of them here. The compiler also has several, more minor problems—for example, it ignores the last line of a file if it isn't terminated with a new line. An *#endif* without a *CR* gives an "unexpected end of file" message. The library function *fgets()* doesn't have this bug in it—why does the compiler?

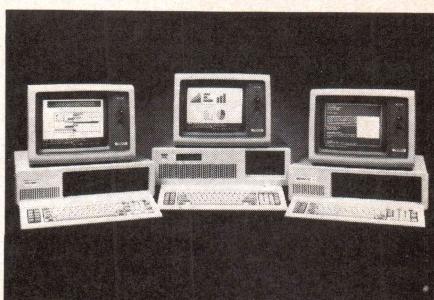
Though the library documentation is nicely done, the *User's Guide* is awful. It has very little discussion of unfamiliar, Microsoft-specific parts of the language. This lack of detail is made more intolerable because the *User's Guide* spends thousands of words talking about topics with which most readers will be familiar. For example, on page 25 it says: "It is recommended that you create a separate directory for each type of file....(Refer to your MS-DOS documentation if you are unfamiliar with the procedures for setting up and maintaining directories)." Does Microsoft seriously believe that a C programmer doesn't know how to set up a directory? Another example is on page 36 where we find a long treatise on how to create a batch file: "You can create an MS-DOS batch file to set up the compiler environment and invoke the compiler. Creating and using batch files is discussed in full in your MS-DOS manual. This section is intended simply to demonstrate possible uses of the MSC command in a batch file..." and on, and on, and on. The manual talks about how to use batch files for the rest of the page and most of the next. Most readers don't need to be told what a batch file is ad nauseam; those who don't know can find the material in the DOS manual. In fact, the *User's Guide* could be shortened by about 50 percent if it started out with the sentence: "This manual assumes that you are familiar with MS DOS. If you are not, please read your MS DOS documentation before proceeding." Why make readers wade through pages of useless verbiage to find the few buried pieces of real informa-

tion and then not provide them with adequate information when a new topic is introduced?

Worse than this garbage is the lack of real information on many obscure or hard-to-understand parts of the Microsoft C implementation. The keyword *far*, for example, is used to create a 32-bit pointer in a small model program; but the user's manual gives you almost no information, and no examples whatsoever, on how this keyword should be used. I tried for about half an hour to declare a *typedef* for a far pointer to an unsigned object before I hit on the right combination of keywords. The manual was no help and neither were the error messages, most of which said "syntax error" and little else. I don't mind Microsoft's introducing a new keyword, especially one as useful as this one, but it should be documented up the wazoo. It is not.

One final problem and then I'll quit. It's impossible (or rather, so hard that I don't want to bother) to make either ROMable code or .COM files with the Microsoft compiler. In the course of my normal consulting work, I have to write a lot of ROMable code. To do this with most compilers, you need a copy of the C root module so that you can modify it to do your system initializations. You also need a special root module if you want to make .COM files. Lattice and Aztec both provide the root module source with their compilers. Microsoft does not. When I called Microsoft to ask about getting a copy, the staff were pretty reluctant even to talk about it. Finally I managed to talk to Sandra Jacobson, the VP in charge of C product marketing, and she told me that the root source was available but that it cost \$500 (that's five hundred—two zeros after the five). This left me speechless for a moment—\$500 is more than the compiler costs. For \$500 Manx gives you two compilers (one optimizing, the other not); the complete source code for the entire I/O library, not just the root module; copies of several Unix utilities (*grep*, *ls*, *make*, *diff*, and a version of the vi editor); an assembler; a linker; and several other utilities. Where does Microsoft get off asking \$500 for 100 lines of code, without which their compiler is useless to anyone who has to put a program into ROM? The

IBM COMPATIBILITY at a not so IBM price



TECH PC/AT \$1999

PRICE INCLUDES:

- 6MHz 80286 CPU
- 512K
- One, 1.2 MB Floppy Drive
- 8 Expansion Slots
- 195 Watt Power Supply
- Complete MS DOS, PC DOS, Xenix Compatibility
- Runs Lotus 123, dBase III Framework and all other popular AT Software.
- ONE YEAR WARRANTY!!

OPTIONS:

- Tech PC/AT with 20MB Hard Disk \$2399

- Tech PC/AT with 20MB Hard Disk, Monochrome Monitor, Hercules® Compatible Mono/Graphics Card \$2599

Also available with 6-8 MHz Switchable CPU, Tape Backups, Modems, Large Hard Disks, and Networking Systems.

TECH TURBO PC/AT \$2399

6-8 MHz Switchable 80286 CPU

TECH TURBO PC/XT \$1099

PRICE INCLUDES:

- 4 to 7 MHz Software Switchable CPU
- 640K
- Two, 360K DS/DD Floppy Disk Drives
- 8 Expansion Slots
- 135 Watt Power Supply
- ONE YEAR WARRANTY!!

OPTIONS:

- Tech Turbo PC/XT with 20MB Hard Disk \$1699

- Tech Turbo PC/XT with 20MB Hard Disk, Monochrome Monitor and Hercules® Compatible Mono/Graphics Card \$1899

TECH PC/XT \$799

PRICE INCLUDES:

- 4.77 MHz CPU
- 256K
- Two, 360K DS/DD Floppy Drives
- 8 Expansion Slots
- 135 Watt Power Supply
- ONE YEAR WARRANTY!!

OPTIONS:

- Tech PC/XT with 20MB Hard Disk \$1399

- Tech PC/XT with 20MB Hard Disk, Monochrome Monitor, Hercules® Compatible Mono/Graphics Card \$1599

TELEX: 272006

Answer Back-TECH

FAX: 714/556-8325

Visa, MasterCard, Check Accepted

TECH PC PERSONAL COMPUTERS

714/754-1170

2131 S. HATHAWAY, SANTA ANA, CA

92705

©1985 TECH PC

"Hercules" is a registered trademark of Hercules Computer Technology.

*IBM, IBM PC, XT, and AT are registered trademarks of International Business Machines Corp.

Circle no. 245 on reader service card.

C CHEST

(Continued from page 21)

only response I got was that not many of its customers require it. Well, I require it. The company wasn't sympathetic. Then it said that another reason for not releasing the root module is that it didn't want to provide user support for it—the module was "too complex" and Microsoft "wasn't too proud of the way that the program looked." To

my mind, this doesn't speak well for the quality of Microsoft's programming or of its support. If Microsoft can't explain the internal workings of its own compiler, who can? It seems that it'll only provide support if your question isn't too hard. To all appearances Microsoft has gone the way of several large corporations: "We don't care, we don't have to."

In conclusion, the Microsoft C compiler is a good implementation of the C language, with a very complete I/O

library and laudable library documentation. It generates compact code and is easy to use (once you wade through the *User's Guide*). As far as I can tell, the code it produces is error-free. Because of serious flaws in the error recovery mechanism, however, an extremely poorly written *User's Guide*, and a certain reticence on Microsoft's part to really support its own product, the compiler is useless to anyone who is not already a very experienced C programmer and to anyone who has to generate ROMable code or do serious systems programming. Though the library documentation is well organized, it is opaque, not telling you anything about the internal workings of the routines. There's no way (short of disassembling the library) to make ROMable code or .COM files with the Microsoft compiler. Even if you're an experienced programmer, I'm not sure you'd be willing to put up with the awful error recovery, which is bound to slow up your development time, or spend inordinate amounts of time trying to figure out how a library routine works and what undocumented bugs it contains.

As I write this, a copy of the Manx Aztec C 86 compiler has just arrived in the mail and a copy of Lattice, Version 3, is on the way. I'll report back sometime in the next few months, after I've had a chance to use them for a while.

Shell Availability

This column is part of a four-part series describing the entire shell. A reprint of all four parts along with a disk containing the listings and an executable version of the shell are available for \$29.95 from *Dr. Dobb's Journal*, 2464 Embarcadero Way, Palo Alto, CA 94303. Please see the advertisement on page 71.

Note

1. Kernighan & Ritchie, *The C Programming Language* (Englewood Cliffs, N.J.: Prentice-Hall, 1978), 183.

DDJ

(Listings begins on page 56.)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 1.



**Greenleaf
Software, Inc.**

2101 Hickory Drive
Carrollton, TX 75006
214/446-8641



**Free CompuView Software!
Call (313) 996-1299 for details.**

VEDIT PLUS®

Multiple File Editor

Word Processor

RECOMMENDED

Come and get it. The editor you've heard so much about. The editor that has been recommended to you for the last five years by BYTE, InfoWorld, Microcomputing, PC, Programmers Journal, Sextant, EDN, Jerry Pournelle, Peter Norton and numerous other reviewers.

'VEDIT is a lightning fast text editor with all the commands of a slick word processor...a text oriented programming language enables you to perform tasks impossible with a standard word processor. A fantastic product.'

The Whole Earth Software Catalog, 1985.

'VEDIT is fast, functionally rich and configurable to your whims. Its programming ability lets its usage stretch as far as your imagination will allow.'

The OMNI Complete Catalog of Computer Software, 1985.

Now there's new VEDIT PLUS. It does it all. Program development. Document preparation. Word Processing. Convert WordStar files, edit dBASE source files, process mainframe files. Whatever your application, VEDIT PLUS can handle it.

VEDIT PLUS is the choice of thousands of engineers, writers and programmers who demand the most powerful text editing software.

Expect more from VEDIT PLUS. It's available for all MS-DOS, PCDOS, CP/M-86 and CP/M-80 computers. List price \$225. Educational / corporate site licensing and current user discounts available.

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. WordStar is a registered trademark of MicroPro, International. dBASE is a registered trademark of Ashton Tate. MS-DOS is a registered trademark of Microsoft. CP/M is a registered trademark of Digital Research.

VEDIT PLUS FEATURES

- Simultaneously edit up to 37 files of unlimited size. Do 'cut and paste' operations, edit text or develop macros.
- 'Virtual' disk buffering simplifies editing of large files.
- Memory management supports up to 640K
- MS-DOS, PCDOS pathname and CP/M user number support.
- Horizontal scrolling (edit spreadsheets easily).
- Customization - determine your own keyboard layout, support any screen size, any computer, any CRT terminal

EASY TO USE

- Interactive on-line help. Create your own on-line help with menus for compilers, style guides, non-computer subjects.
- On-Line integer algebraic calculator.
- 'Undo' command.
- Single key search and selective replace. Search with wild cards and pattern matching.
- Keystroke macros - create your own on-line editing functions.
- Print any portion of text or file.
- Excellent indexed documentation. Includes new tutorial.
- Highly optimized for IBM PC, PC XT or PC AT.

PROGRAM DEVELOPMENT

- Automatic Indent/Indent for 'C', PL/I or PASCAL.
- Conditional lower to upper case conversion. Change at ';' or other character.
- Optional 8080 to 8086 source code translator macro.

WORD PROCESSING

- Word wrap and paragraph formatting at adjustable margins.
- Right margin justification.
- Recognizes graphic, foreign and special character sets.

MACRO PROGRAMMING LANGUAGE

- 'IF-THEN-ELSE', looping, conditional branching, user prompts, keyboard input, algebraic expressions and variables.
- Simplifies complex text processing, formatting, conversions and translations.
- Edit multiple files automatically - perform numerous search/replace in several files without user intervention.
- Complete TECO capability.
- Free Macros: • Compare two files and merge work done by two people • Sort mailing lists • Print formatter • Replace command prompt with an easy to use menu.

A Variable Metric Minimizer

by Joe Marasco

This article describes a variable metric minimizer, a program that finds the set of parameters to a function that will produce the smallest output value from that function. Minimization is a common problem in many fields. A familiar example of this process is the least squares fitting of a set of data to a straight line. Here, an explicit set of equations is used to derive the optimal value for the slope and the intercept of a straight line that approximates the input data. (The minimized function is the sum of the squares of the deviations of the data points from the straight line.) That is, least squares fitting finds the equation of the line that goes as closely as possible through the middle of a set of data points. Of course these points have to be linelike to begin with; forcing randomly scattered data into a line has no meaning.

The least squares function, because it's using lines, requires you to find two parameters (the slope and the intercept). To fit a parabola rather than a line to your data, you'd have to find three parameters instead of two. In general, polynomial fitting for a degree n polynomial requires you to find $n+1$ parameters.

But you need not limit yourself to polynomials. Your data might fit a more exotic combination of powers, such as a rational fraction or a continued fraction. It might also

The program does unconstrained minimization in n dimensions by looking at the global topology of the problem as well as local information from the derivatives.

contain terms with trig, log, or other transcendental functions. In general, we'd like to be able to find the minimum for any function, regardless of the complexity. All data can't be meaningfully depicted as a straight line.

When no limits are

put on the parameters to a function, we have an "unconstrained" minimization problem in n dimensions (where n is the number of parameters that can vary). Imagine an n -dimensional surface; our problem is to find the "lowest" point on the surface. As you might expect, this process can be both difficult and time consuming. The "search space" becomes very large as the dimension of the problem increases. By the late 60s and early 70s, the general minimization problem was brought under control on large mainframe computers for functions of about 10–20 parameters. Today methods exist for handling much larger problems, but these are not really amenable to microcomputer implementation.

The program presented here uses methods now accepted as both robust and efficient. That is, these methods can handle a wide variety of functions and data without failing, and they can find a minimum in a shorter period (fewer iterations) than other methods. Some methods seek minima by a sort of "curve crawling." Like the proverbial drop of water, they tend to smell "down" and go in that direction.

The steepest descent method, for example, looks at the local topology (read: derivative information) and points itself orthogonally (at right angles) to the level curves in an effort to point toward the minimum. The process is

Joe Marasco, Billybob Software, P. O. Box 363, Belmont, CA 94002-0363

© 1985, Billybob Software. All rights reserved.

clarified by thinking of a contour map of a volcanic crater. Just below the rim, the contour map will show a series of concentric circles. The program looks at this map, points itself toward the center by finding a direction at a right angle to a contour line, and off it goes. There are other considerations in a real problem. For example, how big a step do you take? (You don't want to go up the other side.) You get the idea though. The spacing of the contour lines will help you with step size.

Of course this example is a bit contrived. Many geometries aren't concentric circles. The contour lines may often look like wickedly curved and contorted bananas, making the program waste a lot of time going toward the minimum. Think of a meandering river: It's always going downhill. At each point it is locally following the largest gradient. Nevertheless, when viewed from above, the river doesn't follow the shortest path from the top to the bottom.

The methods I have used are called "quasi-Newton positive definite secant update" methods. The "variable metric" part comes from looking at the global topology of the problem in addition to the local information given by the derivatives. That is, we're looking at the river both from above and from the perspective of the river itself. These methods try to take bananalike contours and transform coordinates internally so that they appear to the algorithm as concentric circles. Because the "metric" for doing this transformation changes as we move about, the methods are called variable metric methods. These methods converge more rapidly to the minimum than do other approaches.

Davidon seems to have been the first person to hit on these methods. The Davidon-Fletcher-Powell (DFP) method is still one of the best algorithms available. A similar method—the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method—is thought to be somewhat superior for some problems. These two methods have survived lots of empirical testing, and most people consider them to be the best. Both methods are used in the accompanying program as well as a "steepest descent" method for comparison's sake.

A word of caution: All these methods have some limitations. They all tend to find a local minimum, but over the hill there may be a deeper valley. Your local minimum may not be the global minimum. Hence these methods may depend somewhat on where you start them. The better your original estimate of where the minimum might be, the better chance you have of not getting stuck in the wrong one when there are multiple minima. Unfortunately in higher dimensions it's often difficult to guess how many minima there might be or where they might be. Saddle points can be troublesome too, fooling the program into thinking that it has arrived at the minimum. This difficulty is compounded as you use higher dimensions.

Theory

This section describes briefly how the minimizer works so that you can follow the code. The description is pretty meaty; you may want to skip past it if the mathematics of the problem don't interest you. On the other hand, if you'd really like to understand the inner workings of the mathematics, any of the references at the end of this article will be more than sufficient.

The program loop is divided into four main parts: The first uses the current approximation to the inverse Hessian matrix (this contains what I've previously called the global topology information). The current values of the derivatives of your function with respect to each of the parameters are also used to obtain a search direction. As long as the matrix remains positive definite, the theory guarantees that we will always point downward. So, our minimum will decrease at each step if nothing goes awry.

In the second part, the step size is optimized with a linear search along the direction chosen in part 1. A reasonable trial step size is attempted, and the value of the function and its derivatives are calculated for the new location. Once again, if everything works, this trial step brackets the true minimum. A cubic interpolation between the previous position and the trial position will find our new minimum. Although this search doesn't have to be exact, we can run into problems here. The system will quit if the cubic interpolation fails because of strange geometries; if the minimum was not bracketed, a step size larger than the trial may be attempted. In any event, the result of this step is a new proposed minimum.

In the third part we decide whether or not we've finished. If the new minimum isn't lower than the old one, we stop because the rounding off is probably killing us—the new minimum is always supposed to be lower. (Incidentally, this means you cannot go out of a valley to a lower valley unless something unusual happens.) We'll also quit if the estimated distance to the minimum, as computed internally by the program, is smaller than the first stopping criterion.

We stop if the percent change in the minimum (from the previous value) is less than the second stopping criterion. No matter what happens, we stop if we've used up the number of iterations allowed by the user. If we decide to exit, the reason for stopping is returned to the main program.

Should we make it through part 3, we prepare for another iteration. Here, derivatives of the function at the new minimum are computed (so that they'll be ready for part 1 on the next go-round). The inverse Hessian matrix is updated here too, using either the Davidon-Fletcher-Powell or Broyden-Fletcher-Goldfarb-Shanno updating method. Both of these methods guarantee that the inverse Hessian will remain positive definite, so the new minimum will indeed be lower than the old one. If the steepest descent is being used, the update is skipped because the identity matrix is always used in this case. The update completed, we return to part 1 above for another spin through the loop.

As you've probably noticed, derivatives are computed twice per iteration in this scheme, a penalty of all methods that include a linear search. When minimizing using experimental data sets with many points, the computation of the derivatives is by far the most time-consuming process. Each time we evaluate the derivative vector, we need to make $2n$ function calls, where n is the number of parameters in the function. The number of function calls per iteration is approximated as $4n + 2$. Anything you can do in your function to speed things up will have a great effect on total computation time. The other computations take much less time.

A final word on the derivatives—there are $2n$ function evaluations per derivative because the derivatives are computed by a simple symmetric finite difference. The function is computed once after adding an epsilon to the value of the parameter in question. It's computed a second time after subtracting an epsilon. Then we take the difference, dividing by twice epsilon and holding all other parameter values constant.

Note that if the epsilon is too small, the derivative will be imprecise (because we may be subtracting two numbers that are almost equal and then magnifying the difference by dividing by a small number). On the other hand, if the epsilon is too big the approximation of the derivative—the secant instead of the tangent, as it were—will be too gross. This problem has no easy solution. The default is set to something reasonable, and the derivative calculation can be examined by setting "debug" to 3 (more on this in a moment). You can then set derivative step sizes individually for each parameter on input. After that you have to fend for yourself.

The Program

The program I have just described solves only small- to medium-size problems; it's dimensioned to handle up to ten parameters, although you can adjust this number. To complete the program, you need to add a function representing your model, compile that function, and link it with the other modules. The load module then does the minimization, subject to various input commands you give it. Actually the design of the system is predicated on building a library of models so that various optimizations can be performed against sets of data—more on building this library in a moment.

My version of the system was built on an Osborne computer (running CP/M 2.2) using Software Toolworks's C/80 compiler (Version 3.0) with floating-point support. Digital Research's relocating assembler, library manager, and linker were also used. All these products were up to the task, although a special tip of the hat goes to Walt Bilofsky's folks at Software Toolworks for an exceptional product.

Most of the C code should port easily to other machines. The file global.h that is included in all source files contains conditional compilation directives to ease the porting task. For example, the Software Toolworks compiler doesn't support the type *double*, so I've included a *#define* for the C/80 compiler to change all *doubles* to *floats*.

If you're using the C/80 compiler, leave the *#define C80* statement in global.h. Otherwise, remove it to get the standard *#includes* for stdio.h and math.h. I used some of Software Toolworks's library functions: *atoi*, *atof*, *fabs*, *strcmp*, *strcpy*, *strlen*, *isspace*, *fopen*, and *fclose* are Unix-like and should pose no problems. I've assumed that *isspace()* is a subroutine rather than a macro, however. If *isspace()* is defined by

```
#define isspace(c) ((c) == ' ' || (c) == '\t' || (c) == '\n')
```

the invocation

```
isspace(*p++)
```

is expanded to

```
(*p++ == ' ' || *p++ == '\t' || *p++ == '\n')
```

where *p* is incremented three times if it's not a space, tab, or new line.

Two other portability problems are resolved by the *#define C80* statement—the use of *getline* instead of *fgets* and the need to use *%lf* rather than *%f* in *scanf* calls when using *doubles* instead of *floats*.

Lint is unhappy with one part of VMM.C. The problem has to do with passing around pointers to functions. VMM.C needs to read a string containing a function name, look up that function in a table, and then pass the address of the function back to the calling program. So, we have to declare a function that returns a pointer to a second function that, in turn, returns a *double*. The correct declaration syntax is:

```
double (*obj(a,b))()
double a, b;
{
    ...
}
```

The argument list has to be in the innermost set of parentheses. Unfortunately, the normally stalwart C/80 doesn't accept this declaration. There is a convenient, though very nonportable, way to work around it, though. In fact, some compilers won't accept the code as given in the listings. I assumed that an *int* and pointer are the same size (a valid assumption in an 8080) and then wrote the code accordingly. Listing Nine (next month) shows the essentials of the VMM code as it stands. Listing Ten (next month) is a short sample program that shows the changes you have to make for the program to be correct.

It's obviously to your advantage to use an 8087 if you have it; the calculations in the program are floating-point intensive.

As I mentioned earlier, you'll have to provide a model of your function as a C subroutine. I've provided several templates that you can look at to see what this model should look like. You must respect the number and type of all arguments to the function call; even if you don't use all of them, they all must be included for consistency. The variable *user* (in *main()*) lets you communicate with your function from the outside; when the minimizer calls your function, it calls it with *user* set to 0.

You must update the function library routine (*funlib()*) in VMM.C, Listing Two, page 76 every time you add a function to the system. You can find detailed instructions for updating *funlib()* in the comments in the code. Briefly, a few tables need to be changed. The system chooses a function interactively at run time, and the tables serve to automate the function selection process. These tables tell the program what the function is called when input is being interpreted, what name it has in the system, how

Now dBASE is bilingual.

Announcing a second language
for dBASE®.

C.

Now you can add richer, faster features to the dBASE you know and love with "dBASE Tools for C™".

So you can continue to program in the dBASE programming language, and yet have state-of-the-art calc speed and unique fast-painting graphics.

Here's your tool kit:

A basic engine that links C, special C libraries, and your own C functions to dBASE applications. (It supports Lattice® C, Microsoft® C, and Manx Aztec™ C.)

Arrays management and a C library of financial, mathematical, and statistical functions come with the Programmer's Library.

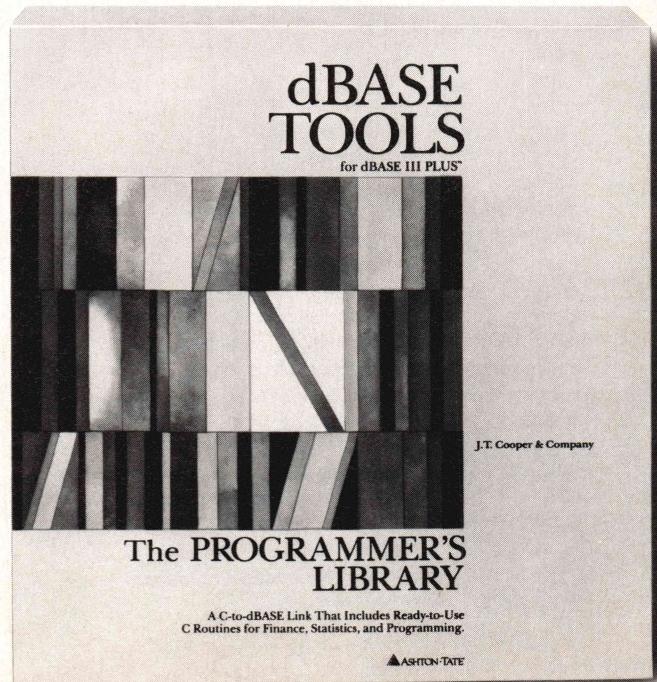
And the Graphics Library includes interactive business graphics

like bar graphs, pie charts, exploded pie charts, marked point graphs, line charts, and XY charts.

To order, for the name of your nearest dealer, or for more information, call the Ashton-Tate Publishing Group at 800-437-4329, Ext. 242.

Sure you need C to be on the leading edge.

But you don't have to give up dBASE to get it.



Trademarks/owner: Ashton-Tate, dBASE/Ashton-Tate. Microsoft/Microsoft, Inc.; Lattice/Lattice, Inc.; Aztec/Manx Software Systems Inc.
© 1986 Ashton-Tate. All rights reserved. Specifications subject to change without notice.

Circle no. 242 on reader service card.

METRIC MINIMIZER

(Continued from page 26)

many parameters it has, how many constants it has, and what names you have chosen for the parameters.

Adding a function is easy. By grouping the main program, the function library routine, and all the user functions in VMM.C, all the things that you're likely to change are isolated in one module, so VMM.C is the only module that ever needs to be recompiled. To rebuild the system, edit VMM.C, recompile it alone, and then link it to the remainder of the system. You may want to put the other modules into a library to ease the linking process.

Before modifying the system, it's best to get it working and play with it for a while. Compile and link all the modules provided, and then run the program using the sample data file I've provided (a1.dat, Listing Eight, next month). Unix-style redirection can be used to both enter the data and put the results into a file:

```
A>VMM <A1.DAT >A1.RES
```

You can interactively input the commands found in a1.dat too and see the results directly on your screen.

The Program's Output

Some of the program's behavior deserves comment. There are several levels of debug printout, so you can have the program glide through to the result quickly and silently or you can see it work in three levels of detail. Silent operation is important if the minimizer is embedded into another application. Sometimes, however, it is important to see what the program is doing. Debug level 1 prints out some summary results at each iteration. Level 2 displays details of how the program is making its internal computations and decisions. Level 3 goes one step further and provides details of the derivative calculations.

You can set the debug level for each data case as part of the data input process.

bfgs	select BFGS method
constants	set constants in function
debug	set debug level
derivsteps	set step size for derivatives
dfp	select DFP method
end	exit program
epsilons	set stopping criteria
exmin	set expected minimum
funname	select function (req'd first)
iterlim	set maximum iteration
itermin	set minimum iteration
limitflags	set limits on parameters
lowerlimits	set lower limits
newdata	read a data file
next	do this data set, then read next
pstart	set starting values for parameters
reset	set reset value on update matrix
sd	select steepest descent method
upperlimits	set upper limits

Command Input

Another area of interest is the command input process. I've tried to make the program easy to use without robbing it of its flexibility for advanced users. Most of the important parameters can be adjusted on input, although a case can be run with as few as three commands: one to name the function to use, one to give the starting values for the parameters, and one to invoke the calculation for that case. Multiple cases can be stacked in one run. Each of these cases is referred to below as a data set.

As you can see from the code (and from the sample data in Listing Eight), input is formatted as a keyword followed by either zero or more numeric values or strings. If the first word on the line is not recognized as a keyword, the entire line is taken as a comment. This is convenient for embedding comments in the output listing as all input commands are echoed, but might cause trouble if you're not paying attention when entering data. Recognized keywords are shown in Table 1, below.

The first command in any data set must be *funname*. This command should be followed by the name of the function to be looked up in the library. (The table that you set up in *funlib()* tells the program how many parameters and constants this function requires.)

The only other mandatory command is *pstart*, which takes as arguments n floating-point numbers corresponding to the starting values for the n parameters expected by the function named *funname*.

The minimization algorithm can be selected with one of the *sd*, *dfp*, or *bfgs* commands. These correspond to steepest descent, Davidon-Fletcher-Powell, or Broyden-Fletcher-Goldfarb-Shanno methods. The *dfp* command is used by default.

The minimum and maximum number of iterations are specified with the keywords *itermin* and *iterlim*, each followed by a fixed-point number. If these commands aren't used, defaults of n and 2n (where n is the number of parameters) are assumed.

The *reset* command tells the program to reset its update matrix periodically after a specified number of iterations. If you don't issue a *reset* command, a default of 3n/2 is used.

Stopping criteria can also be specified. By default, if the estimated distance to the minimum becomes less than 1.0e-06 or the fractional change in the minimum becomes less than 1.0e-03, the program will stop (provided that the minimum number of iterations has been executed). You can alter these limits with the keyword *epsilons* followed by the limit on the estimated distance to the minimum and the limit on the fractional change (in that order). Both numbers must be entered, even if you want to change only one of them.

One or more parameters can be constrained to a specified range using the *limitflags*, *lowerlimits*, and *upperlimits* keywords. The three must always be used together. *Limitflags* is followed by n integers set to either 0 or 1, turning limit flags off or on. If the flag is set to off (the default if a command is not used), then there are no constraints on that parameter. The keywords *lowerlimits* and *upperlimits* must be followed by n floating-point values for the lower and upper limits on each parameter; the values for those parameters with flags set to 0 are

Table 1: VMM keywords

SAS Institute Inc. Announces

Lattice C Compilers for Your IBM Mainframe

Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

Today...

Our efforts are complete—we have a first-rate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

■ Generation of reentrant object code.

Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.

■ Optimization of the generated code.

We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.

■ Generated code executable in both 24-bit and 31-bit addressing modes.

You can run compiled programs above the 16 megabyte line in MVS/XA.

■ Generated code identical for OS and CMS operating systems.

You can move modules between MVS and CMS without even recompiling.

■ Complete libraries.

We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library

routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix®-style I/O access method.

- **Built-in functions.** Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix link-editor and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

For more information...

Complete and mail the coupon today. Because we've got the development tool for your tomorrow.



SAS Institute Inc.
SAS Circle, Box 8000
Cary, NC 27511-8000
Telephone (919) 467-8000 x 7000

I want to learn more about:

- the C compiler for MVS software developers
- the C compiler for CMS software developers
- the cross-compiler with PLINK86 and PLIB86

today...so I'll be ready for tomorrow.

Please complete or attach your business card.

Name _____

Title _____

Company _____

Address _____

City _____ State _____ ZIP _____

Telephone _____

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000, Cary, NC, USA.
27511-8000. Telephone (919) 467-8000, x 7000

METRIC MINIMIZER

(Continued from page 28)

ignored. An example of the process is given at the top of Listing Eight. Constraints on parameters are very useful when, for example, you know that a certain parameter must be positive (it might be a physical dimension). In this case you can set its lower limit to 0.0 and its upper limit to some large number. You can also fix a parameter by setting its limit flag to 1 and setting its lower and upper limit to its starting value (this particular use of limits is rare but may prove useful). Constraining parameters slows down the system significantly as there are several places where a trigonometric transformation must be employed to convert from a bounded interval to an infinite one and back again.

The *derivsteps* keyword lets you set the size of the interval used for computing the derivative. The default interval is 1 percent on either side of the current value.

The keyword *exmin* adjusts the expected value of the minimum (0.0 is the default).

To allow you to find best sets of parameters for whole families of functions, your function may use several constants, set using the keyword *constants* followed by m floating-point values for the m constants that your function requires. Constant values are remembered from data

8 bits (Osborne I)

single precision

Z-80, CP/M 2.2

5 1/4-inch floppy

Scale	Method	Result	Minimum	Iterations
Starting values: (-1.2, 1.0, ..., -1.2, 1.0)				
100	sd	-.67140, .45982	14.	4
	dfp	-.860, .749	17.	7
	bfsgs	-.84, .71	17.	7
10	sd	.889, .788	6.1 E-2	20
	dfp	.99, .99	7.4 E-5	20
	bfsgs	.98, .97	1.5 E-3	20
1	sd	.96809, .92871	5.5 E-3	20
	dfp	.99982, .99964	1.6 E-7	10
	bfsgs	.99996, .99994	8.3 E-9	10
Starting values: (1.2, 0.8, ..., 1.2, 0.8)				
100	sd	.95171, .90560	1.2 E-2	10
	dfp	.9825, .9651	1.6 E-3	9
	bfsgs	1.0036, 1.0072	6.8 E-5	8
10	sd	.96279, .92363	7.5 E-3	20
	dfp	.9986, .9971	1.0 E-5	7
	bfsgs	.99842, .99678	1.3 E-5	7
1	sd	.99944, .99869	1.7 E-6	10
	dfp	.99986, .99971	1.0 E-7	6
	bfsgs	.99983, .99963	1.6 E-7	6
Time (total) 670 seconds				
Time per run 37.2 seconds				
Time per iteration 3.2 seconds				

Table 2: 8-bit benchmark

set to data set as a convenience. Nevertheless, if you change functions you must include a new *constants* statement to avoid using the old constants.

Debug sets the debug level; follow it by 0, 1, 2, or 3. Note that the debug diagnostics start being printed as soon as this command is issued, so putting it in at the start of a data set gives you debug diagnostics as part of the input echo. Debug level 0 (no diagnostics are printed) is the default.

Use the *newdata* command to read a file with experimental data. Follow the command with the full file name for the data (e.g., *b:exp.dat*). The data file is assumed to have a very specific format: The first element is the number of lines to read from the data file (extra lines are ignored). The rest of the file is data (in floating-point format). The x value must be entered first, then the y value. The data is put into an array of structures called *exp* (defined near the top of Listing Two). If you need something different (such as three numbers instead of two), you'll have to change the definition of the DATA structure (in *global.h*), as well as the associated input routines, and then recompile. Note that the maximum number of data points is defined as *DMAX* in the main program and will probably need to be enlarged. If you want to echo the data as it's read, turn on the debug before requesting *newdata*. The program aborts if it can't open the data file.

One caveat with *newdata*: Data read into the array persists from one data set to the next (in the same way as do constants). It didn't make sense to have to reread the data from disk for each new case if the same data was being used. Anyway, if you want to change data files from one case to another, you must issue a call to *newdata* with a different file name. Incidentally, I used the keyword *newdata* instead of something such as *data file* to emphasize that you are in fact reading in new data.

The *next* command starts up the algorithm (moves you from a Data Input to a Run mode) for the current data set. With the exceptions of data entered with *newdata* and constants read in with *constants*, each set starts with a clean slate—there's no carry-over from the previous data set. If debug was set to 1 on the previous data set, and you still want it to be 1, you must request it again.

The keyword *end* causes the program to terminate.

If an error is detected in a data set, the set is skipped. If the reason for the data error isn't obvious, the case should be run a second time with debug set to 1 at the very start of the case. You can always reset it to 0 just before issuing a *next* command, thereby getting additional debug printout just for the input stage. Sometimes the error is caused by a mistyped keyword, which causes the whole line to be treated as a comment. If debug is set to 0, there's no indication that this has occurred. The program always tells you how many commands and how many comment lines it's read, however, so you have a quick way of checking for this problem even if debug is off.

Benchmarks

I ran two benchmarks to show the relative performance of the various algorithms mentioned earlier (see Tables 2, left, and 3, page 32). The Rosenbrock function:

$$F = K * (x_2 - x_1^2)^2 + (1 - x_1)^2$$

**Another in a series of
productivity notes on
MS-DOS™ software
from UniPress.**

**Subject: Multi-window full screen
editor.**

Multiple windows allow several files (or portions of the same file) to be edited simultaneously. Programmable through macros and the built-in compiled MLISP™ extension language.

Features:

- Famed Gosling version.
- Extensible through macros and the built-in compiled MLISP extension language.
- Dozens of source code MLISP functions; including C, Pascal, LISP and MLISP syntax checking.
- EDT and simple WordStar™ emulation modes.
- MS-DOS commands can be executed with output placed in an EMACS window.
- Run a compile on your program and EMACS will point to any errors for ease of debugging.
- EMACS runs on the IBM-PC™ (XT/AT), TI-PC™, DEC Rainbow 100+™, HP-150™ or any other MS-DOS machine. Requires at least 384K.

Price:

EMACS binary	\$325
EMACS source	995
One month trial	75

Also available for UNIX™ and VMS™
Call for pricing.

**UNIPRESS
EMACS™**

Subject: Compiler for MS-DOS

Lattice C Compiler is regarded as the finest compiler for MS-DOS and produces very efficient and compact code.

Features:

- Runs on the IBM-PC under MS-DOS 1.0, 2.0 or 3.0.
- Produces highly optimized code.
- Small, medium, compact, and large address space models available.
- Full C language and standard library.
- 8087™ floating point support.
- PLINK™ linkage editor is optionally available to support modular programming.

Price:

Lattice C Compiler	\$425
PLINK	395

**COMPILER
FOR THE 8086™ FAMILY**

**LATTICE®
C
COMPILER**

**Subject: UNIX-like "make" facility
now for MS-DOS.**

Ps-Make is a powerful programming aid providing time saving steps for the programmer by compiling only changed components of a program.

Features:

- Ps-Make compiles only changed components of your program. There is no need to execute long batch files that re-compile your entire system.
- Ps-Make directly executes the compiler, linker and other utilities, or automatically generates a batch command file containing only those commands that must be executed to bring the program up-to-date.
- Uses standard UNIX "makefile" syntax.
- Ps-Make can be used with any compiler, including Lattice C.
- Two modes: Batch requires 64K, Direct requires 256K.

Price:

Ps-Make	\$90
---------	------

For our **Free Catalogue** and more information on these and other software products, call or write:
UniPress Software, Inc.,
2025 Lincoln Hwy., Edison, NJ 08817.
Telephone: (201) 985-8000.

**Order Desk: (800) 222-0550
(Outside NJ). Telex 709418.**

European Distributor: Modulator SA,
Switzerland Telephone: 41 31 59 22 22,
Telex: 911859

OEM terms available.
Mastercard/Visa accepted.

*See our ad in the next issue for
more MS-DOS products, including
PHACT™ and PCworks™.*

**UNIX "MAKE"
FACILITY**

PS-MAKE

UniPress Software

Trademarks of UniPress EMACS/MLISP UniPress Software, Inc. MS-DOS, Microsoft IBM PC, IBM Corp. WordStar MicroPro Int'l Corp. TI-PC, Texas Instruments, UNIX, AFT Bell Laboratories VMS/DEC Rainbow 100+, Digital Equipment Corp., Lattice, Lattice, Inc., HP 150, Hewlett-Packard Co., PLINK Phoenix Computer Products, Corp. 8086/8087 Intel Corp. PHACT, PHACT Associates, PC works, Touchstone Software Corp.

METRIC MINIMIZER

(Continued from page 30)

was used. F is 0 when

$$x_2 = x_1^2 \text{ and } x_1 = 1$$

So, at $x_1 = 1$, $x_2 = 1$, we have a minimum for all $K > 0$.

The constant K allows us to "scale" the two terms; for $K = 1$, the two terms have about the same contribution to F in the neighborhood of the minimum. As K becomes large, the problem becomes progressively more difficult: The first term, which "drives" x_2 to be x_1^2 , will swamp the second term, which drives x_1 to 1. For $K = 100$, we would expect to see x_2 approach x_1^2 but not see x_1 go to 1. I have included tests with $K = 1, 10$, and 100 on an extended Rosenbrock function using ten instead of two variables; that is, the function is made up of ten terms consisting of five pairs of variables related in the same way. The tests have been done using a poor starting point $(-1.2, 1.0)$ and a fairly good starting point $(1.2, 0.8)$ so you can evaluate the importance of a good starting value.

All tests were performed using the same maximum number of iterations (20). The inverse Hessian approximation was reset every five iterations in the *dfp* and *bfgs* methods. The factor K is called scale in the tables.

16 bits (Intel 86/30 System)

double precision

8086/8087, Xenix

25 Meg hard disk

Scale	Method	Result	Minimum	Iterations
Starting values: $(-1.2, 1.0, \dots, -1.2, 1.0)$				
100	sd	.67131, .45970	14.	4
	dfp	.64703, .42826	14.	7
	bfgs	.63495, .41267	13.	7
10	sd	.89008, .78805	6.1×10^{-2}	20
	dfp	.9983, .996	1.3×10^{-5}	20
	bfgs	.999, .998	8.1×10^{-6}	20
1	sd	.96809, .92871	5.5×10^{-3}	20
	dfp	.99980, .99961	2.0×10^{-7}	10
	bfgs	.99980, .99960	2.0×10^{-7}	10
Starting values: $(1.2, 0.8, \dots, 1.2, 0.8)$				
100	sd	.95171, .90559	1.2×10^{-2}	10
	dfp	.981, .9624	1.8×10^{-3}	10
	bfgs	1.0031, 1.0062	4.8×10^{-5}	10
10	sd	.96279, .92363	7.5×10^{-3}	20
	dfp	.99846, .99668	1.5×10^{-5}	5
	bfgs	.99837, .99650	1.6×10^{-5}	5
1	sd	.99943, .99872	1.7×10^{-6}	10
	dfp	.99986, .99970	9.7×10^{-8}	5
	bfgs	.99982, .99962	1.6×10^{-7}	5
Time (total)			81 seconds	
Time per run			4.5 seconds	
Time per iteration			0.4 seconds	

Table 3: 16-bit benchmark

The correct result is $(1.0, 1.0, \dots, 1.0, 1.0)$, which would be indicated below as 1.0, 1.0. The number of digits in the result shows how much the answer varies from pair to pair among the five pairs of results. The minimum should be 0.0.

The Listings

Several listings are provided with this article. Global.h (Listing One, page 74) is the global *include* file mentioned earlier. It contains conditional compilation switches for the C/80 compiler versus other C compilers.

VMM.C (Listing Two) is the main program, the function library module, and all the functions in the library. For the time being, there are three functions: *cohen()* is the test problem in Cohen's book (see the reference in code), *sine()* finds coefficients for a polynomial approximation by fitting sine data, and *rosen()* is used as a benchmark. You'll have to compile, assemble, and then link this file to add a function to the library. All the other files may be kept in object format and just linked to VMM.OBJ (or .O, .REL, or whatever) to generate a new load module.

MINIM.C (Listing Three, page 82) contains the main minimizer routine.

AUXLC.C (Listing Four, next month) contains the line search procedure and the decision logic for stopping the minimization.

UP.C (Listing Five, next month) holds the routines required for updating the "topology matrix" according to the DFP or BFGS methods.

UTIL.C (Listing Six, next month) contains a variety of utility routines. Among these are routines for transforming from external to internal coordinates when dealing with constrained variables, routines for calculating derivatives, routines for initializing everything at the beginning of a data set and for setting proper defaults, vector and matrix output routines, and routines for various matrix and vector manipulations.

READ.C (Listing Seven, next month) is the input interpreter. There are a few conditional compilation directives in this module so that *scanf* will handle *floats* and *doubles* correctly.

A1.dat (Listing Eight) is a sample input for a test run, including the Rosenbrock benchmark mentioned above. A1.dat reads in sine.dat (Table 4, below), which contains the experimental data for the function *sine()*.

Availability

This program is put in the public domain for personal,

20	.050000	.078459	.550000	.760406
	.100000	.156434	.600000	.809017
	.150000	.233445	.650000	.852640
	.200000	.309017	.700000	.891006
	.250000	.382683	.750000	.923879
	.300000	.453990	.800000	.951056
	.350000	.522499	.850000	.972370
	.400000	.587785	.900000	.987688
	.450000	.649448	.950000	.996917
	.500000	.707107	1.000000	1.000000

Table 4: sine.dat

END RUN dBASE III[†]

The dBASE[†] sign-off has never made more sense.

Because Word-Tech software can run your dBASE programs 3 to 10 times faster. Run them on multi-user and networking systems.

Eliminate site licenses and runtime fees forever. And save you over \$500 on a single-user, extended version of dBASE III.[†]

Our MS-DOS compilers run your dBASE applications faster and in just 128K of free memory on any MS-DOS computer, not only the IBM PC.

You don't pay any penalties because WordTech compilers use the same language and syntax as dBASE and the same index, memory and data files (up to 10 data files open simultaneously, each with up to 7 indexes).

And WordTech dBASE II[†] and III compilers are just \$750, once. Then you only need a single copy of dBASE for every programmer, not

Good idea.

every user, and can distribute as many copies of your compiled (and protected) programs as you want with no strings attached.

And now you can use dBASE III on multi-user and networked systems. Our multi-user compiler runs your existing dBASE programs under AT&T's UNIX System V and CROMIX. And our networking compiler runs your applications on DOS 3.1

local area networks (LAN's). The compiler will take care of file and record locking for you, or you can take full control using dBASE III networking commands.

We can extend your applications—and your budgets—even further with dBFrame™ for windows, dBChart™ for business graphics, and dB-XL™, the \$169 interpreter that replaces dBASE III.

And we back everything with free support, one year of maintenance and a money-back guarantee.

For details and the name of your nearest dealer, contact WordTech Systems, Inc., P.O. Box 1747, Orinda, CA 94563. (415) 254-0900. TELEX 503599.

It might be a good idea to call now.



**WORDTECH
SYSTEMS**

METRIC MINIMIZER

(Continued from page 32)

nonprofit use only. Permission is granted to reproduce it, but I would appreciate your commenting any changes you make. As the publicly available source code is obviously beyond my control, I can assume no liability for subsequent versions.

This source code is available on several CP/M remote BBSS, in particular on the Mountain View RCPM (415) 965-4097 and the First Osborne Group (FOG) RBBS #1 (415) 755-2030. Both of these systems provide free downloads to members satisfying nominal entry requirements.

I plan to keep a list of registered users in order to support continuing development and evolution of this system. Registered users participate in a common pool of bugs and bug fixes as well as potential future updates. To become a registered user, send a check for \$25 to: Billybob Software, P.O. Box 363, Belmont, CA 94002-0363.

A registered user can obtain the source code on a 5 1/4-inch disk by sending a preformatted disk and sufficient return postage to the above address. Be sure to indicate what format your disk is expecting. The most recent version will be returned. I can handle many CP/M formats as well as PC DOS and MS DOS formats; if you require an unusual format, send a postcard and I'll let you know if I can accommodate you.

Bibliography

There are lots of good books in this field, and the list given here could easily be three times longer. I've tried to include those books that specifically address the minimization problem.

Acton, F. S. *Numerical Methods That Work*. New York: Harper & Row, 1970. One of the best overall treatments of minimization, there are both wisdom and technique in this book. It's a good place to start if you're new to numerical analysis.

Cohen, A. M. *Numerical Analysis*. New York: Halstead Press, Wiley, 1973. This book is a paperback, inexpensive but solid. Cohen's notation is used throughout the program described in this article. He describes the Davidon-Fletcher-Powell method only.

Dennis, J. E., and Schnabel, R. B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs, N.J.: Prentice-Hall, 1983. A definitive work in the field but it's very theoretical.

DDJ

(Listings begin on page 74.)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 2.

Frustrated With the Tyranny of PASCAL? Tired of the Drudgery of BASIC?

Free Yourself With CCSM, the Database Language...only \$59.95

Compare This Routine to Your Present Language, and See the Difference

```
RD  READ "NAME: ",NAM, ! QUIT:NAM=""  
IF NAM'2.A1","1A.E WRITE "PLEASE ENTER AS LAST, FIRST MI",! GO RD  
TEL  READ "TEL # ",TEL, ! IF TEL'?3N1"-^4N WRITE "NNN-NNNN PLEASE",! GO TEL  
SET ^DATA(NAM)=TEL GO RD  
PRT  WRITE " NAME",?20,"TELEPHONE #",! SET NAME=""  
LP   SET NAM=$ORDER(^DATA(NAM)) QUIT:NAM="" WRITE NAM,?20,^DATA(NAM),! GO LP
```

This simple program accepts, screens and saves names and phone numbers...sorts and prints them. These six lines of code are an example of the extremely compact, and familiar nature of COMP Computing Standard MUMPS, the Database Language. In lines 1 and 2, READ, IF, WRITE and GO should be easy to follow. The pattern match operator "??" filters for the correct input of alpha characters to make a name. In line 4, SET ^ DATA creates a permanent global file, with NAM as a subscript. The data node is SET to the telephone number. In line 6, the \$ ORDER command gets the next subscript in order, from the ^ DATA file, thereby SETting NAM to the next name in the file.

CCSM, the Database Language, frees you from the tyranny of typed and restrictive languages...NO declarations of variables or data files. Look at these Features:

- Full Screen Editor
- Virtual Memory (routines and variables may be as large as a disk)
- Multi-User available...up to 15
- B-Tree File Structure
- 8087 and BCD Support
- Exceeds 1984 ANSI Standard MUMPS
- Transportable from Micro to Mini to Mainframe

CCSM, the Database Language, is a fast, modern version of ANSI Standard MUMPS, developed by COMP Computing. It comes with a 20 year history of development, solving database applications. CCSM improves programmer productivity, and efficiency...typical programs are written in 1/3 the code of BASIC or PASCAL. CCSM is an easy to learn language and comes with a 250 page manual.

AMEX, VISA and
MASTERCARD
accepted by phone.

1-800-257-8052

In Texas 713-529-2576

CCSM, the Database Language, sells for \$59.95, and comes with full documentation. Until March 31, 1986, for an additional \$15.95, we'll send along the "Cookbook of MUMPS", and its disk, (reg. \$24.95) containing useful routines and utilities. For charts and graphs, order the Graphics disk for \$49.95. Multi-user version, \$450. Disks are non-copy-protected. Requires IBM PC or compatible with 128K. (Macintosh version available...\$89.95)

Order by phone, or clip and mail:

1-800-257-8052

in Texas, 713-529-2576

AMEX	— card no. _____	\$75.90
VISA	— _____	\$59.95
MC	— CCSM, Cookbook, and disks special package CCSM, the Database Language Graphics disk Please add \$3.00 for shipping and handling. Texas residents add 6 1/8% sales tax.	\$49.95

name _____

street _____

city _____ st _____ zip _____

IBM PC and Macintosh are trademarks of International Business Machines, and Apple Computer.

Circle no. 98 on reader service card.

THINK

FAST!

If you're a C programmer
you could be a more productive C programmer.
Introducing Lightspeed C™ from THINK Technologies, Inc.

Lightspeed C is a compiled programming environment for the Macintosh™ that gives you speed, convenience, and top quality code generation, too.

With Lightspeed C, turnaround is 1000% faster. Time to build from scratch is 3 times faster. Time to link a typical 15,000 line program is 5 seconds.¹ And generated code quality is better than any on the market.

Best of all, Lightspeed C's, integrated Edit-AutoMake-Launch environment makes turnaround a one-step process.

If you want to produce higher quality results with less time and effort, send for Lightspeed C today.

¹The above statements are based upon benchmarks for creating an executable version of XLISP from scratch and by modifying, re-compiling, and re-linking one source file. Comparisons against Mac C™, Aztec C™, and Megamax C™ were performed using a 512K Macintosh™ with a 10MB Hyperdrive™.



Send me Lightspeed C™ fast. Name _____

I need more to think about,
send me information about
Lightspeed C™

Title _____ Company _____

Address _____

City _____ State _____ Zip _____

Telephone _____

MC Visa Amex Signature _____

Acct # _____ Exp. Date _____

To order: Please enclose check for \$175.00 for each compiler.

Mail to:

THINK Technologies
420 Bedford Street
Lexington, MA 02173
Or call 617-863-5595

Concurrency and Turbo Pascal

by Ernest E. Bergmann

An appropriate use of coroutines is in monitoring the keyboard and the serial port under a terminal program.

After much trial and error, I have developed a method to use Turbo Pascal (CP/M-80) to implement several concurrent tasks. These techniques are useful for writing software that must respond to several real-time events, such as a modem program. In this article, I will explain what concurrency is, what it is useful for, and how to achieve it. Turbo Pascal provides a convenient way to become acquainted with multitasking because of the ease of writing and trying variations quickly and clearly.

Subroutines vs. Coroutines

A subroutine or function call is frequently used in programming to carry out a "subtask" that a main task needs in order to continue. Economy of coding occurs when this same subtask is needed in several different places. The subtask is subservient to the main task in that its existence depends upon being called upon by the main task.

A pair of routines should appear to proceed in random order, namely the starting or completion of either routine is not in complete lockstep with the other routine. Doing several things at the same time is often called multitasking; it is an important feature of the languages Modula and Ada.

An example of a possible use of coroutines is the implementation of a terminal program on a personal computer. In this case, one task has the responsibility of monitoring the keyboard. Every time a key is pressed, the corresponding character is sent out of the serial port. The other task monitors the serial port

for incoming characters, and when one arrives, it is sent to the display. These two tasks are more or less independent as they do not need to share any function.

An important characteristic of the operation of such a program is that you do not assume that the arrival of characters at the serial port is closely related to tapping the keyboard. Failure of characters to arrive from either source should not interfere with the operation of the other task.

A skeleton of the program should be:

```
task1;
begin
  repeat
    serout(keyin);
    until false(forever);
  end;
task2;
begin
  repeat
    videout.serin);
    until false(forever);
  end;
```

Each of the tasks resembles the main procedure of a Pascal program. As is good form in structured programming, all the tough work gets put off to be written in the subrou-

tines and functions (particularly *serin* and *keyin*). Before I discuss the subroutines in detail, note that, for the tasks to run independently of each other, they need to have separate stack spaces. The stack is used to save return addresses (among other things) when subroutines and functions are called. If both tasks shared a common stack, there would be an undesirable tendency for return addresses (and possibly data) to get confused between the two tasks. Thus, each task should be assigned its own stack.

In Listing One, page 88, you can see that *task1* and *task2* indeed start by assigning their own stack spaces. Although *mystack* is declared as a variable in both *task1* and *task2*, the tasks are assigned separate memory locations by the Turbo Pascal compiler. The compiler assigns different locations for every variable of the main program and for every parameter, return value, and variable of every function and procedure.

The two functions *keyin* and *serin* are characterized by taking an unpredictably long time to complete their tasks—for example, *keyin* could take hours if the keyboard operator fell asleep (and meanwhile other things might be happening). These two functions consist of a loop where the function (tediously) asks whether a character is available. It is in these loops that most of the time is wasted; as soon as a character becomes available, the function can return its value.

The crux of accomplishing concurrency is to share the processing power between tasks. This sharing is accomplished by the calls to *defer* done by *keyin* and *serin*. *Defer* is treated as a procedure, but it succeeds in switching the processor from one task to another (hence the choice of

Ernest E. Bergmann, 730 Seneca Street, Bethlehem, PA 18015

name). It should be invoked wherever a procedure might not proceed quickly. *Defer* should also be invoked from time to time if rapid response to external events (such as key presses) is needed and a procedure or function is doing extensive calculations.

I found it convenient to treat the main portion of the program as *task0*; it is special in that it is given initial control when the program starts. It must initialize variables for *defer* to operate; other initializations can be done here as well, such as sending commands to hardware to configure baud rates and so on. In addition, the way the program should terminate is through the main procedure; the Pascal compiler does not empower the subroutines to terminate the program in a normal manner. To help in start-up and termination, two procedures, *initall* and *exit*, are provided.

The coding of *defer* was arrived at after some experimentation. I found, for example, that assigning an array element with the hardware stack pointer (*stackptr*) results in a different value than assigning a simple variable with *stackptr*. The reason is that the compilation generates code that pushes an address pointer for the element on the stack and so affects the value of the stack pointer at that moment.

When *defer* is called, the return address in the calling task is on the (current) stack. When *defer* switches stack pointers (thus switching to the stack of another task), it will return to the point in the new task where *defer* had been called.

A task is entered at two places: at the start and reentry (from where it had been deferred). It is necessary for *defer* to know which (re-)entry to use. This is accomplished with a test of the saved stack pointer. If the saved value is zero, then it can assume the task has never been entered and it should start at the beginning. When a task is started, it should immediately initialize the stack pointer to its own stack space.

In the interests of clarity and easy modification, the scheduling aspect of *defer* has been broken off into a separate procedure. The functioning of *schedule* here is almost trivial; it is apparent that for special applications with more tasks, you might

wish to try other scheduling algorithms than round robin.

The beauty of using the procedures *initall*, *defer*, *schedule*, *exit*, and the like is that they may be used as black boxes—the details of how they work can be ignored in their use. Once I had written them, I concentrated my efforts on coding each individual task.

Reentrancy

Emboldened by my success with us-

ing Turbo Pascal to implement multiple, independent tasks, I went on to try multitasking with procedures that are shared among several tasks. My first attempts failed miserably, and I learned the hard way that procedures and functions in Turbo Pascal are not reentrant even if they may be recursive.

A requirement of Pascal is recursion, which is supported by Turbo Pascal (using the {\$A-} option). Recursion is where procedures or func-

C-terp

**The C
Interpreter
You Won't
Outgrow**



C-terp will grow with you as you progress from novice through professional to guru. Unbelievable, but true, the easiest-to-use C interpreter will provide you with the most advanced programming features for upward growth. Our exclusive **object module support** enables you to add libraries (like HALO, PANEL, Windows for C, etc., or your own homebrew libraries) to C-terp as you add them to your computing repertoire. Use C-terp as a microscope on your libraries! Flip a bit and allow our **software paging** (NEW) to handle those big jobs! There are no fixed-size tables to overflow, and C-terp can be configured for different screens and screen adapters (NEW). With multiple modules and **full K&R support**, we offer a dream C environment.

- Our new improved **configurable editor** competes with anything going.
- Speed -- Linking and semi-compilation are breathtakingly fast.
- Convenience -- Errors direct you back to the editor with the cursor set to the trouble spot.
- Symbolic Debugging -- Set breakpoints, single-step, and directly execute C expressions.
- Compatibility guaranteed -- batch file to link in your compiler's entire library. Supported compilers include: Computer Innovations C86, Lattice C, Microsoft C 3.0, Mark Williams C86, and Aztec C.
- Many more features including batch mode and 8087 support.

What Our Users/ Reviewers Are Saying

- "... easy to use, powerful, and a timesaver."
- "... we absolutely LOVE C-terp."
- "... has restored my faith in interpreters."
- "... a programmer's dream."
- "... wonderful technical assistance."
- "... increased our productivity by a factor of 40."
- "... the best C product ever, in any category."

- Price: \$300.00 (Demo \$45.00)
MC, VISA

Prices include documentation and shipping within U.S. PA residents add 6% sales tax.
Specify compiler.

- C-terp runs on the IBM PC (or any BIOS compatible machine) under DOS 2.x and up with a suggested minimum of 256 Kb of memory. It can use all the memory available.

* C-terp is a trademark of Gimpel Software.

GIMPEL SOFTWARE

3207 Hogarth Lane • Collegeville, PA 19426
(215) 584-4261

CONCURRENCY

(Continued from page 37)

tions call themselves directly or indirectly (by calling other procedures or functions that call them). An example of recursion would be the implementation of *autoCR*. *AutoCR* precedes a line feed with a carriage return:

```
procedure putc(c: char);
begin
  if c=LF then putc(CR);
```

end;

You cannot assume that all languages support recursion—for example, FORTRAN does not. To support recursion it is necessary that each invocation of a procedure has its own distinct variable space. If the space is not unique (if recursion is not supported), then in the above example the value of *c* would be changed by the second invocation of *putc* to a

CR and in all likelihood, the *LF* (the original invocation) would not be printed but replaced by a *CR*; two *CRs* would be produced instead of the desired *CR-LF* sequence.

Although Turbo Pascal supports recursion when you use the *{\$A-}* option, it does not support reentrancy. For a procedure to be reentrant, it must be possible to defer in the middle of an invocation, switch to another task, and for that other task to reenter the same procedure without affecting the subsequent operation of the first invocation of that procedure. It was assumed in the development of Turbo Pascal, I presume, that it would not be used in a multitasking environment.

The most straightforward way around this difficulty is not to require reentrancy. Any procedure or function that might be deferred directly or indirectly and that is called by more than one task should be duplicated as many times as is necessary so that each copy is used by only one task. Listing Two, page 90, is an example of this approach.

Listing Two is a program I developed to demonstrate the use of queues to enable tasks to operate more independently of each other. In operation, it displays the characters typed at the keyboard on the screen. What is noteworthy about this program is that it is quite easy for the typist to type faster than the display rate and, thus, to "type ahead." The display continues to show more and more characters even after the typist has paused; no characters have been "lost" by the slowness of the display relative to the keyboard.

To make things more interesting, I provided the XON/XOFF protocol to make the display stop altogether on demand and to resume on demand later. If you were to type a Ctrl-S then the display would not show any activity even though you might be typing another sentence or two. When you eventually type a Ctrl-Q, the display will reactivate and show whatever was held back.

As a touch of friendliness, you can correct your typing by typing a backspace to undo the last character typed. An additional convenience is that the program provides line feeds after any carriage return you type to put you on the next line without

Tools for the Programmer from Blaise Computing

Save Up To \$130 On These Special Offers!

TOOLS & TOOLS 2

For C or Pascal

For a limited time, pick up both packages and save \$50 off our regular list price. The C version comes with libraries for the Lattice, Computer Innovations and Microsoft (version 2.03 and

3.00) compilers. The Pascal version supports IBM and Microsoft Pascal.

\$175.

VIEW MANAGER With Source

All libraries are included. Please specify C or Pascal. Regular \$425. Save \$130.

\$295

Blaise Computing provides a broad range of fine programming tools for Pascal and C programmers, with libraries designed and engineered for the serious software developer. You get clearly written code that's fully commented so that it can serve both as a model and also be easily modified to grow with your changing needs. Our packages are shipped to you complete with comprehensive manuals, sample programs and source code. None of the programs are copy-protected.

FOR C AND PASCAL PROGRAMMERS:

TOOLS ◇ \$125

Extensive string and screen handling, graphics interface and easy creation of program interfaces. Includes all source code.

TOOLS 2 ◇ \$100

Memory management, general program control and DOS file support. Interrupt service routine support. Includes all source code.

VIEW MANAGER ◇ \$275

General screen management. Create data entry screens that can be easily manipulated from your application program. Block mode data entry and retrieval with fast screen access.

VIEW LIBRARY Source ◇ \$150

Source code to the VIEW MANAGER library functions.

ASYNCH MANAGER ◇ \$175

Powerful asynchronous communications library providing interrupt driven support for the COM ports. All source code included.

FOR THE TURBO PASCAL PROGRAMMER:

Turbo POWER TOOLS ◇ \$99.95

Extensive string support, extended screen and window management, interrupt service routines, program control and memory management, interrupt filters. All source code included.

Turbo ASYNCH ◇ \$99.95

Interrupt driven asynchronous communication support callable from Turbo Pascal. ASYNCH is written in assembler and Turbo Pascal with all source code included.

PACKAGES FOR ALL PROGRAMMERS:

EXEC ◇ \$95

Program chaining executive. Chain one program from another even if the programs are in different languages. Common data area can be specified. Source code included if you're a registered C TOOLS and C TOOLS 2 user.

SPARKY ◇ \$75

Run-time resident (or stand-alone) scientific, fully programmable, reverse polish notation calculator. No limit on stack size, variables or tape. Includes all standard scientific functions and different base arithmetic.

TO ORDER, call Blaise Computing Inc. at (415) 540-5441

◆ 2034 Blake Street ◆ Berkeley, CA 94704 ◆ (415) 540-5441

BLAISE

Watch us!
BLAISE COMPUTING INC.

Circle no. 159 on reader service card.

your having to remember to type the line feed as well. Should you type a line feed, it will be discarded.

Listing Two consists of three tasks. The first task (*keyboard*) gathers whatever is typed and quickly stores it away so that it can be digested at leisure without missing any keystrokes. The method of storage is known as a queue, reminiscent of people standing in a line in which the first person in line is the first to get service. Another expression for the same technique is FIFO (first in, first out). No one seems to use the acronym LILO (last in, last out), although it probably would be understood equally well. In any case, the first task places all characters received into the first queue by calling the procedure *enqueue1*.

The second task is called the *filter*. Its function is to take each character that it obtains from the queue of characters received by *task1* and either pass them along or carry out particular actions. If a carriage return is encountered, you want a line feed to follow it. If a line feed is typed, it is to be dropped. Ctrl-S and Ctrl-Q characters are not passed along but are used to stop and start the output, respectively. A Ctrl-C causes the program to quit. Most characters are passed along.

The output of the *filter* does not go directly to the output device but is placed in a second queue awaiting a chance to go to the display whenever the display is ready to take this output. Thus the *filter* produces output by calling the function *enqueue2*.

The third task, *printer*, takes characters from the second queue by calling *dequeue2*. *Dequeue2* either returns with a character right away or may take its time (deferring to other tasks). Delays in returning a character mean that there are no characters (and you need to wait for the *filter* to supply some more) or that the queue is in a "noflow" state. Because the video display is normally very fast, I chose, for purposes of illustration, to slow the printer task artificially by requiring it to count up to PRATE before sending each character.

Notice that the functions *occupancy(p)* and *vacancy(p)* are each used by more than one task. *Task1* uses *vacancy* (through *enqueue1*). *Task2* uses *occupancy* (through *dequeue1*) and

vacancy (through *enqueue2*). *Task3* uses *occupancy* (through *dequeue2*). The functions *occupancy* and *vacancy* can be "shared" between several tasks because they never defer.

On the other hand, I have learned the hard way not to try to share *enqueue(p)* and *dequeue(p)* because these functions would defer, and they need to be reentrant. Thus I have separate but equal procedures: *enqueue1* and *enqueue2*. Also, I have the separate functions *dequeue1* and *dequeue2*.

Conclusion

It is relatively easy to create a multitasking program with Turbo Pascal in CP/M-80 using the method I have shown here. You have to be careful, though, that functions or procedures that need to be reentrant are replaced by several copies to avoid that need.

DDJ

(**Listing begins on page 88.**)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 3.

DISCOVER THE LANGUAGE OF ARTIFICIAL INTELLIGENCE **PROLOG V**

Interpreter for MS-DOS/PC-DOS

Not Copy Protected



At last! A Prolog with enough muscle to handle real-world applications for UNDER \$100! Discover why Japan has chosen Prolog as the vehicle for their "Fifth Generation Machine" project to design intelligent computers.

CHOOSE FROM TWO GREAT VERSIONS:

PROLOG V-Plus

\$99.95

- More Than 100 Predefined Predicates
- Large Memory Model (to 640K)
- Floating Point Arithmetic
- 150-Page User's Manual and Tutorial plus Advanced Programming Documentation
- Co-Resident Program Editor
- Calls to Co-Resident Programs
- Text and Graphic Screen Manipulation

PROLOG V

\$69.95

- 70 Predefined Predicates
- Small Memory Model
- Integer Arithmetic
- 122-Page User's Manual and Tutorial

FREE FREE FREE FREE FREE
Programming in Prolog
by Clocksin & Mellish \$17.95 value
Available with purchase of PROLOG
V-Plus only. Offer valid through
March 31, 1986

STANDARD FEATURES ON BOTH:

- Clocksin & Mellish-Standard Edinburgh Syntax
- Extensive Interactive Debugging Facilities
- Dynamic Memory Management (garbage collection)
- Custom-Designed Binder and Slipcase

THE CHOICE OF UNIVERSITIES

Generous university site licenses and an excellent teaching tutorial and reference guide have made PROLOG V the choice of universities nationwide. Call for details.

■ ■ ■ PHONE ORDERS: 1-800-621-0852 EXT 468 ■ ■ ■

PAYMENT ENCLOSED \$ CA residents add 6% sales tax

CHARGE MY: MasterCard Visa

Card No. _____ Exp. Date _____

Signature _____

Mr. / Mrs. / Ms. _____ (please print full name)

Address _____

City / State / Zip _____

PROLOG V-Plus \$99.95
PROLOG V 69.95
UPGRADE ONLY 40.00
Return factory diskette and
\$30 plus \$10 Handling

SHIPPING:
\$ 5.00 U.S.
7.50 Canada
10.00 Caribbean,
Hawaii Air
20.00 Overseas Air

COD Orders Not Accepted
15 day check clearance

Upgrade
to PROLOG V-Plus
for only the difference
in price plus a
handling charge.

NO RISK OFFER
Examine the documentation
at our risk for 30 days. If
not fully satisfied, return
with disk still sealed for
full refund.



CHALCEDONY
SOFTWARE

5580 LA JOLLA BLVD.
SUITE 126 D
LA JOLLA, CA
92037
(619) 483-8513

Circle no. 178 on reader service card.

The Problems of Parallelism

by Michael Swaine

Parallel processing involves both a re-thinking of system architecture and a fundamental recasting of algorithms. It's a Knuth 50 task.

Life has a grammar with four terminal symbols. G, T, A, and C (guanine et al) are four chemical bases that combine to form the two-stranded, helical molecules of DNA, the grammar behind all life on Earth. In pairs, the bases form sequences up to hundreds of thousands of base pairs long that define the genetic makeup of an organism. Several thousand such functional sequences of base pairs have been identified, but they represent only a small fraction of the possible sequences. Having learned that structural similarity is the first clue to functional similarity, molecular biologists such as J. Douglas Welsh at Princeton University begin the process of identifying the function of each new sequence by comparing, base pair by base pair, the new sequence with existing sequences. Unraveling the thread of life is a problem in pattern matching.

Many good algorithms for pattern matching are known, but good algorithms are not always good enough. Welsh found that the problem was using more computer time than he could afford, and he worried that the pruning heuristics he employed to focus only on "relevant" sequences would cause him to overlook important functional sequences. He wanted to search both exhaustively and efficiently. Existing algorithms and architectures seemed to prevent him from doing so.

Those who read the discussion of the Fgrep algorithm in *DDJ* (September 1985) will recall how algorithms that do pattern matching in parallel can sometimes outpace their sequential counterparts. Princeton graduate student Daniel Lopresti, enlisted to

Michael Swaine, 2464 Embarcadero Way, Palo Alto, CA 94303

solve Welsh's problem, followed a similar insight and found a parallel algorithm that he estimates can produce a thousandfold increase in processing speed over the algorithm Welsh had been using. Lopresti has implemented his algorithm in a multiprocessor chip that represents an advance in the technology for molecular biology and could also find applications in many other disciplines in which string comparisons are important.

Lopresti's example shows that replacing traditional sequential algorithms with parallel algorithms can produce significant increases in throughput and efficiency. In fact, problems that have been shown to be NP-complete and to require exponential time to solve by sequential methods can often be solved in polynomial time, and sometimes in linear time, using parallel techniques. The development of appropriate parallel algorithms for fundamental problems such as searching, sorting, pattern matching, matrix operations, fast Fourier transforms, and so on is by no means a simple matter, however. Like Lopresti, the designer of a successful parallel solution to a programming problem today is likely to

be programmer and microprocessor designer in one body because implementing a parallel solution tends to involve the entire computational model, and the parallel algorithm can rarely be viably grafted onto an existing sequential Von Neumann architecture.

Concurrency

There are, of course, many examples of concurrent processing in otherwise sequential architectures. Concurrency at the level of the operating system in the case of Concurrent DOS or at the "operating environment" in the cases of TopView and DESQview, for example, allows entire applications to run in what is, at that level, parallel fashion. At a deeper level, the machine is executing instructions in strictly sequential fashion. Taking it a step higher, one can implement concurrent processes within an application program; IBM claims that TopView will support concurrency within the programs that it is running concurrently. Another article in this issue examines the techniques for implementing concurrent processes. (See "Concurrency and Turbo Pascal" by Ernest Bergmann.)

There are also examples of deeper concurrency that still don't force any serious rethinking of fundamental algorithms or of the architecture of the machine. Print spooling and the use of dedicated math coprocessors both split off easily segregated operations and hand them off, freeing the CPU for other operations. A more complex separation of functions was presented by James Vaughan and Robert Smith of Tantivy Associates of Palo Alto, California, at the Wescon show last November. Their machine implements a structured sequencer that operates in parallel with arithmetic operations. The sequencer just

traverses a tree; it supplies the calling and branching operations of traditional sequential programming. The computational component is a coprocessor that performs all the traditional arithmetic operations, passing single-bit messages to the sequencer and picking up its instructions and data from it. But the resultant concurrency between sequencer and processor is, as Vaughan himself points out, similar to the kind of concurrency that exists between a main processor and a math coprocessor in a personal computer.

Parallel Processing

True parallel processing involves both a rethinking of system architecture and a fundamental recasting of algorithms. It exists. There are parallel-processing computers in existence today, and more are on the way. Full use of those computers—exploitation of parallel processing at the level at which we have exploited sequential processing today—represents a large step forward, but it's not here yet. It's a level 50 problem in the notation of Knuth's *Fundamental Algorithms*. General-purpose parallelism in machine architecture and algorithm design may well require cooperation on the level of the efforts that produced the sequential machines of the 1940s. That cooperation is not always present.

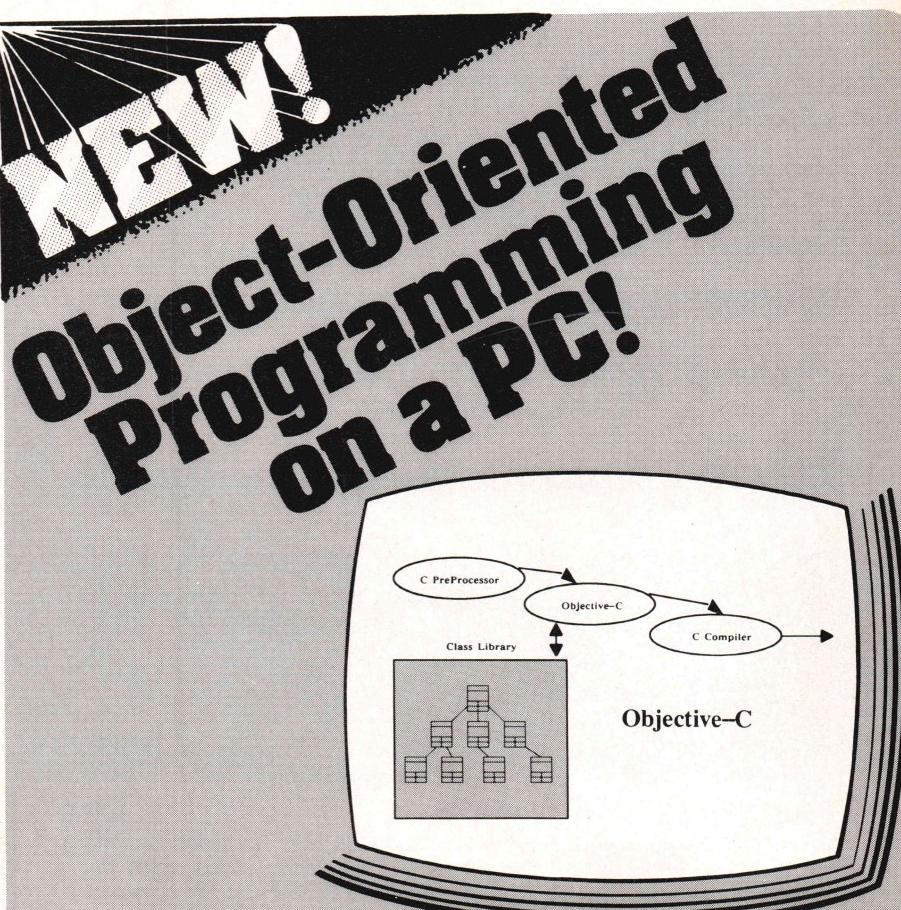
A year ago, Peter C. Patton was growing frustrated directing efforts at the Microelectronics and Computer Technologies Corp. (MCC) in Austin, Texas, to solve the problem of decomposing tasks into components appropriate for parallel execution. After resigning his post with MCC, he expressed his frustration with the organization's lack of focus in *The Wall Street Journal*: "Most people think parallel processing is the future, but nobody can agree on how to get there," he said.

One organization trying to provide some focus is the Parallel Processing Research Council, a group that grew out of a 1983 NSF workshop on parallel-computing research. According to *IEEE Computer* (December 1985), the group plans to promote collaborative efforts in parallel-processing research among government, industry, and university researchers by producing a newsletter, distributing in-

formation about nonproprietary research and projects in parallel hardware and software, and starting two research facilities. A report issued by the council attributes any lack of progress in the area to lack of facilities and significant collaborative effort, deficiencies the group hopes to remedy.

One of the differences among ap-

proaches to parallelism is the architecture vs. algorithm distinction. It stems from the necessity of finding a good match between algorithm and architecture and the question of just how to achieve the match. Does one start with the architecture and tailor the algorithms to it or develop efficient parallel algorithms and create architectures to support them?



Objective-C™ is an object-oriented programming language and a fully documented library of reusable components...adding messages, objects and inheritance to C language. Applications written in Objective-C are fully compatible with other Objective-C compilers running under UNIX, VMS or AOS.

Objective-C provides the productivity of object-oriented programming, while

retaining the portability and efficiency of C. PPI also provides comprehensive technology transfer to insure that your programmers fully understand this exciting new technology.

PPI's Objective-C compiler generates C code, which requires the Microsoft V3 C compiler running under MS/DOS.

At \$500 Objective-C is affordable. Order today!

PRODUCTIVITY PRODUCTS INTERNATIONAL
27 Glen Road, Sandy Hook, CT 06482. (203) 426-1875.



Circle no. 140 on reader service card.

PARALLELISM

(Continued from page 41)

Perhaps the problems have to be solved in parallel.

Architecture-Driven Design

According to Sudhakar Yalamanchili and J. K. Aggarwal of the University of Texas at Austin, writing in *IEEE Computer* (December 1985), the approach of tailoring the algorithm to the architecture can achieve high

performance at the expense of generality. They refer to this approach as architecture-driven algorithm design. In particular, architectures employing different interconnection topologies among processor, memory, and I/O resources will force recasting of parallel algorithms.

Two broad architectural approaches to parallelism are the multiple-instruction stream/multiple-data stream (MIMD) and the single-instruction stream/ multiple-

data stream (SIMD) architectures. Machines embodying each of these architectures will be appearing on desktops within a year. Ncube Corp. of Beaverton, Oregon, will be selling a four-processor card for the IBM PC/AT this year that is mostly intended for use by developers of software for the larger parallel systems the company is planning. Ncube's card employs MIMD architecture.

Apparently an architecture-driven design can at least peak out at good performance figures, even if it is not the optimal parallel solution for all problems. ITT expects to release a SIMD board for desktop computers next year. If estimates by Steven Morton, ITT's designer, are correct, machines with the SIMD board will reach performance levels up to two-thirds of a Cray-1 and cost around \$10,000.

ITT and Ncube are both developing general-purpose machines. On the other hand, there are examples, such as Princeton University's Navier-Stokes machine, of architectures developed to support particular parallel algorithms.

The Navier-Stokes equations for fluid flow are used to solve problems in fluid dynamics. The equations are good candidates for parallel processing, and Daniel Nosenchuck and Michael Littman are building a 128-node computer dedicated to solving Navier-Stokes equations. They developed the parallel algorithm and designed the machine to implement it. They expect to crank through equations at 50 gigaFLOPs, a performance level achieved at some expense in generality.

The algorithm-driven approach, in contrast to the architecture-driven approach, is suggested by Lopresti's DNA machine and by machines running Prolog programs. Prolog suggests some common methods of parallelism, chiefly AND-parallelism and OR-parallelism. The DNA machine's algorithm is specific, while the approaches to parallelism implicit in Prolog are quite general. Many problems, cast as Prolog programs, suggest a natural parallel formulation—natural for Prolog, at least. But despite such general schemes for parallelizing algorithms, the development of efficient parallel algorithms for common problems is still in its early stages.

C Programmer Essentials

"Offers many capabilities for a reasonable price"

W. Hunt, PC Tech Journal

"I highly recommend the C UTILITY LIBRARY"

D. Deloria, The C Journal

C ESSENTIALS

200 functions: video, strings, keyboard, directories, files, time/date and more. Source code is 95% C. Comprehensive manual with plenty of examples. Demo programs on diskette. Upgrade to THE C UTILITY LIBRARY for \$95.

THE C UTILITY LIBRARY

Thousands in use world wide. 300 functions for serious software developers. The C ESSENTIALS plus "pop-up" windows, business graphics, data entry, DOS command and program execution, polled async communications, sound and more.

ESSENTIAL GRAPHICS

Fast, powerful, and easy to use. Draw a pie or bar chart with one function. Animation (GET and PUT), filling (PAINT) and user definable patterns. IBM color, IBM EGA and Hercules supported (more soon). NO ROYALTIES. Save \$50 when purchased with above libraries. Available February, 1986.

Compatible with Microsoft Ver. 3, Lattice, Aztec, Mark Williams, CI-C86, DeSmet, and Wizard C Compilers. IBM PC/XT/AT and true compatibles.

C Compiler Packages: Microsoft C - 319, Lattice or CI-C86 compilers -\$329. Save \$40 - \$50 when purchasing compiler and library combinations. Specify C compiler and version number when ordering. Add \$4 for UPS or \$7 for UPS 2-day. NJ residents add 6% sales tax. Visa, MC, Checks, P.O.'s.

ESI ESSENTIAL SOFTWARE, INC
P.O. Box 1003 Maplewood, NJ 07040 914/762-6605

Circle no. 138 on reader service card.

Communication

As in the 1940s with the development of the sequential computer and its associated algorithms, algorithms and architecture for parallel processing may have to be developed hand in hand, requiring close communication among researchers.

The Computer Measurement Research Facility of the National Bureau of Standards is collecting parallel benchmark programs. The plan is to develop a public repository of information about parallel-computer performance; the Bureau wants programs, written in high-level languages, that measure some aspect of parallel-processing performance. Direct inquiries to:

Computer Measurement Research Facility
Institute for Computer Sciences and Technology
Materials Building MS B364
National Bureau of Standards
Gaithersburg, MD 20899
(301) 921-3274.

Those interested in joining the Parallel Processing Research Council should contact:

George Almasi, PPRC Chair
IBM T. J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
(914) 945-1305
ALMASI.YKTVMX@IBM

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 4.

QUALITY SOFTWARE YOU CAN AFFORD!

NEW Z80 Symbolic Debugger

- Only \$49.95 plus shipping.
- Screen oriented with a simultaneous display of instruction mnemonics, register, stack, and memory values.
- Breakpoints may be set on any combination of fixed memory address, register values and/or memory values.
- Uses Digital Research compatible SYM files.
- Supports Hitachi HD64180.

Relocatable Z80 Macro Assembler

- Only \$49.95 plus shipping.
- 8080 to Z80 Source Code Converter.
- Generates Microsoft compatible REL files or INTEL compatible hex files.
- Compatible with Digital Research macro assemblers MAC & RMAC.
- Generates Digital Research compatible SYM files.
- Conditional assembly.
- Phase/dephase.
- Cross-reference generation.
- Full Zilog mnemonics.
- INCLUDE and MACLIB FILES.
- Separate data, program, common, and absolute program spaces.
- Supports Hitachi HD64180.
- Z80 Linker and Library Manager for Microsoft compatible REL files available as an add-on to Assembler.

ATTENTION Turbo Pascal Users:
Turbo Assembler will generate
Turbo Pascal in-line machine code
include files.

TO ORDER, CALL TOLL FREE:
1-800-367-5134, ext. 804

For information or technical assistance:
(808) 623-6361

Specify desired 5 1/4" or 8" format. Personal check, cashier's check, money order, VISA, MC, or COD welcomed.

PRICE LIST

Z80 Macro Assembler: \$49.95
Assembler, Linker, and Library Manager: \$95.00
Manual Only: \$15.00

Z80 Symbolic Debugger: \$49.95
Manual Only: \$15.00
Assembler, Linker, Library Manager, and Debugger: \$134.95

Include \$5 for shipping and handling.

MITEK

Z80 is a trademark of Zilog, Inc. MAC and RMAC are trademarks of Digital Research, Inc. Turbo Pascal is a trademark of Borland International, Inc.

Circle no. 190 on reader service card.

Brand New From Peter Norton A PROGRAMMER'S EDITOR

that's *lightning fast* with the *hot* features programmers need

only
\$50

Direct from the man who gave you *The Norton Utilities*, *Inside the IBM PC*, and the *Peter Norton Programmer's Guide*.

THE NORTON EDITOR



Easily customized, and saved
Split-screen editing
A wonderful condensed/outline display
Great for assembler, Pascal and C

Peter Norton, 2210 Wilshire Blvd., #186
Santa Monica, CA 90403, 213-826-8032
Visa, MasterCard and phone orders welcome

"This is the programmer's editor that I wished I'd had when I wrote my *Norton Utilities*. You can program your way to glory with *The Norton Editor*."

Peter Norton



Speeding MS DOS Execution

by Gregg Weissman

I was intrigued to read in Dr. Dobb's Clinic (DDJ, September 1985) that he thought "the DOS BIOS doesn't use the ROM BIOS for its disk input, and DOS's BIOS does it better." Because I know that this is not so (DOS does use the ROM BIOS), I thought I would experiment and discover more of the story and the answer to the riddles presented in the column.

DOS definitely uses the ROM BIOS to handle disk I/O. This can be verified by examination of the DOS BIOS code as it resides in memory or by loading the DOS BIOS file (IBMBIO.COM for PC DOS or IO.SYS for MS DOS) into DEBUG and unassembling it. Look for the INT 13 instructions: There is the disk I/O handling. You will not see any of the lowest level DMA and disk-controller handling procedures that you find in the ROM BIOS. It would be against IBM's and Microsoft's software design principles to include such low level machine dependencies in the DOS BIOS.

I used E-X-E Software's DOS2TOOLS System Utilities package to perform timing tests to verify the benchmarks published in the column. This package contains a program to trap DOS functions and time them and another program to produce listings of the results. Five listings are included with this article, showing the results I obtained in performing some of the tests mentioned in the column.

I was able to verify most of the test results, obtaining very similar times even though the tests were performed on an IBM PC/AT and not an XT. Because the limiting factor in I/O would be the disk's physical charac-

When writing your own BIOS disk handlers using interrupt 13h, set the head settle time to 0 for all read operations.

teristics and not CPU speed, the timings would be expected to be similar.

The DOS COPY command in my test (Listing Four, page 101) ran in 9.7 seconds, compared to 9.8 seconds in Cortesi's: close enough to consider the results identical. My interpreted BASIC-A test (Listing Five, page 101) ran in 44.2 seconds of DOS time and 66 seconds total. Cortesi includes no figure for the time spent by DOS on the disk functions only.

In the tests that called DOS from assembly language, Cortesi claims that "block sizes beyond 9K, one cylinder, don't give further improvement." This is not what I found: In my tests (Listings Six, Seven, and Eight, pages 102–104), a block size of 9K, or 2400h, gave a result of 9.7 seconds, while block sizes of C000h through FE00h (49,152 to 65,024 bytes) gave substantially lower times, leveling off at C000h at 7.09 to 7.14 seconds (plus or minus one clock tick).

It could be that the block size at which performance levels off is a function of the disk hardware because I obtained the same results with both PC DOS Versions 2.1 and 3.1 and could verify that the BIOS calling sequences were essentially the same. There is no other way to account for Cortesi's finding that performance

did not improve past the 9K block size.

The Answer to a Puzzle

The reason Cortesi states that the DOS BIOS does not use the ROM BIOS for disk I/O, however, is based on the findings of an (unlisted) assembler program that reads the disk directly through the BIOS interrupt 13h. Because this program was slower than the fastest DOS time, he assumed that DOS does not use ROM. I knew there had to be another answer.

I tried my own assembler routine to read the requisite number of sectors directly, also using the ROM interrupt 13h. This appears in Listing One, page 94. There are only a handful of logical ways to do this, so I suspect my program is very similar to Cortesi's. My program also ran slower than DOS, timing out at 11.42 seconds. Because I knew DOS had to be using BIOS, I coded a short resident module to trap the BIOS interrupt 13h so I could see what DOS was doing as it happened.

What I found is that DOS modifies the head settle time parameter in the table of disk variables that BIOS uses to control the NEC disk controller. The head settling delay is a simple do-nothing loop in the ROM BIOS, giving the read/write head and arm time to settle into position after a seek operation is performed, before the controller attempts to read or write any data. Each time the disk has to move from one cylinder to another, this head settle delay will be taken.

The absolute memory location 0000:0078h contains a pointer to a table of values that users can modify in order to accommodate different disk characteristics. One of the parameters is the head settle delay.

The default head settle time on the

Gregg Weissman, E-X-E Software Systems, 205 E. 78th St., New York, NY 10021

AT is 15 milliseconds when the system is initialized by BIOS, and PC DOS 3.1 resets this to 1 millisecond. On the XT, it defaults to 25 milliseconds; PC DOS 2.1 knocks this down to 15 milliseconds. These head settle times are required when writing disk data, but not when reading, as I discovered.

When DOS reads from the disk, it sets the head settle time in the disk parameter block to 0. When the operation is complete, DOS restores the original value. Therefore, when Cortesi (and I) ran our assembler BIOS tests, the head settle time was set to the original default—in my case 1 millisecond, and in his, possibly up to 15 milliseconds.

Two New Puzzles

If I changed the settle time parameter to 0, as does DOS, my assembler timing went from 11.42 seconds down to 7.9—but this presented two puzzling problems.

The first problem was easy: The new lower times were still greater than the DOS timings—the DOS tests consistently showed times of 7.09 to 7.14 seconds, and those times included all the operating system overhead. With a little thought I realized that the motor start-up time must be considered. The motor starts when DOS executes an *OPEN* function, which I was not counting in my comparisons.

When I started the disk motor before beginning the timing, the time required for the BIOS to read 16 cylinders went down to 6.8 seconds, the time Cortesi reports for the *DOS INT 25* function as the fastest possible (which is true). If you refer to the DOS timing figures, the *OPEN* calls all took about 2 seconds (plus or minus 1 clock tick), so everything fits neatly together.

Pay close attention to Cortesi's phrase "DOS sequential input can reach and sustain an input rate of one track per disk rotation"—naturally the best performance possible. I'll explain how and why this is so and show you how to achieve this yourself.

The attentive reader may already have thought of the second, less trivial problem. The head settle parameter is supposed to indicate the number of milliseconds of delay. Eliminating the delay on the AT by setting the count to 0 instead of 1 should save 1 millisecond per seek-

to-track. The test reads 16 different tracks, so there should be an improvement of 16 milliseconds, not more than 3 seconds.

Further tests indicated that increasing the settling time past 1 millisecond resulted in no further deterioration of performance until the count passed 90, at which time I lost another 3-plus seconds, and so on.

The solution appears easy now, but it took some work to arrive at it.

I pored over the AT BIOS listing to find out if the head settle parameter is used in some other way in addition to the seek operation. The answer is no: There is no other use of this variable other than counting the number of 1-millisecond delays taken immediately after a seek.

0.002258300 ← Settle delay loop time as tested.

Settle Delay, Cnt & Millisecs:			Rslt Times:
SettleCount:	0	Totl:	6.662
Delay time:	0.00	Wait:	6.612
Total calc. delay:			0.000
SettleCount:	1	Totl:	9.786
Delay time:	2.26	Wait:	9.699
Total calc. delay:			0.036
SettleCount:	2	Totl:	9.786
Delay time:	4.52	Wait:	9.661
Total calc. delay:			0.072
SettleCount:	4	Totl:	9.786
Delay time:	9.03	Wait:	9.590
Total calc. delay:			0.145
SettleCount:	8	Totl:	9.786
Delay time:	18.07	Wait:	9.450
Total calc. delay:			0.289
SettleCount:	16	Totl:	9.786
Delay time:	36.13	Wait:	9.160
Total calc. delay:			0.578
SettleCount:	32	Totl:	9.786
Delay time:	72.27	Wait:	8.595
Total calc. delay:			1.156
SettleCount:	64	Totl:	9.786
Delay time:	144.53	Wait:	7.457
Total calc. delay:			2.313
SettleCount:	80	Totl:	9.785
Delay time:	180.66	Wait:	6.890
Total calc. delay:			2.891
SettleCount:	88	Totl:	9.785
Delay time:	198.73	Wait:	6.605
Total calc. delay:			3.180
SettleCount:	90	Totl:	9.786
Delay time:	203.25	Wait:	6.536
Total calc. delay:			3.252
SettleCount:	91	Totl:	12.776
Delay time:	205.51	Wait:	9.490
Total calc. delay:			3.288
SettleCount:	92	Totl:	12.975
Delay time:	207.76	Wait:	9.658
Total calc. delay:			3.324
SettleCount:	120	Totl:	12.978
Delay time:	271.00	Wait:	8.650
Total calc. delay:			4.336
SettleCount:	128	Totl:	12.976
Delay time:	289.06	Wait:	8.376
Total calc. delay:			4.625
SettleCount:	255	Totl:	16.164
Delay time:	575.87	Wait:	7.040
Total calc. delay:			9.214

Table 1: Disk performance by head settle time

SPEEDING MS DOS

(Continued from page 45)

To eliminate any noise in my data caused by losing interrupts during processing and the coarse granularity of the usual 18.2-tick-per-second interrupt clock, I coded a routine to access the AT high-resolution timer, which provides 1,024 interrupts every second. For those who are interested, Listing Two, page 95, is the

program used to access the AT clock.

The high-resolution timings confirmed the other results and also gave a more accurate figure for the delay loop as coded in the BIOS. Although the number of cycles in the delay instructions total approximately 1 millisecond of CPU time, because of cycle-stealing memory refresh and other perturbations, the actual time for one iteration of the loop is actually about 2.25 milliseconds. This

still could not account for the 3.5 seconds of performance improvement.

Next I moved the BIOS code down into RAM so patches could be made and breakpoints set: It was impossible to determine anything more from just timing the disk interrupt. I inserted a patch to time the subroutine that waits for the disk adapter to respond to the last command sent. This routine, called WAIT_INT in the BIOS, just loops until an interrupt is received from the adapter, after a command has been sent to the disk-controller chip.

Results of My Tests

I performed 60 timing runs consisting of 20 different settle time parameters, run 3 times each. In each test I counted the overall time required to read the 16 cylinders and the total time spent in the WAIT_INT routine.

Table 1, page 45, shows the means of the times plotted against the head settle time as specified in the parameter block and as calculated according to my 2.25-millisecond figure for the delay loop.

What is interesting to note in this table is that overall disk performance changes as a step function, not continuously, and the times are constant over a wide range. Note also that the WAIT_INT time decreases as the head settle time increases, so the result is constant except at the breakpoints of 2.26, 205.5, and 575.87 milliseconds of settle time.

Within each range, the longer BIOS delays after a seek, the less time it takes for WAIT_INT to receive the interrupt from the controller, until the delay time passes a certain threshold. What is happening at that threshold? If you examine the differences between each plateau, you will see that each jump in overall time is about 3 seconds. If you divide the differences by the number of seeks (16) then you get a figure (0.18, 0.19, and so on) suggestively close to the time it takes for a 300-RPM drive to make one revolution—0.2 seconds. The second jump in the step function occurs when the settling delay increases from 2 milliseconds to 205 milliseconds, also very suggestive.

From these results, I developed the hypothesis that the overall time is constant within each rotational period of the disk. If a 1-millisecond delay

Settle delay loop time as tested:

0.002258

Settle Count:	Calc'ed 1 Loop (Milliseconds)	Total Timed:	Result Waiting:	Calc'ed 16*Days
0	0.000	6.662	6.612	0.000
1	2.258 *	9.786	9.699	0.036 BREAK POINT
2	4.517	9.786	9.661	0.072
4	9.033	9.786	9.590	0.145
8	18.066	9.786	9.450	0.289
16	36.133	9.786	9.160	0.578
32	72.266	9.786	8.595	1.156
64	144.531	9.786	7.457	2.313
80	180.664	9.785	6.890	2.891
88	198.730	9.785	6.605	3.180
90	203.247	9.786	6.536	3.252
91	205.505 *	12.776	9.490	3.288 BREAK POINT
92	207.764	12.975	9.658	3.324
94	212.280	12.970	9.330	3.396
96	216.797	12.978	9.513	3.469
104	234.863	12.978	9.223	3.758
112	252.930	12.978	8.937	4.047
120	270.996	12.978	8.650	4.336
128	289.063	12.976	8.376	4.625
174	392.944 *	16.166	9.666	6.287 BREAK POINT
255	575.867	16.164	7.040	9.214
260	587.158 *	19.350	9.674	9.395 BREAK POINT
346	781.372 *	22.141	9.283	12.502 BREAK POINT

* Analysis of breakpoints:

Mean	6.662	Delta	Settle	0.000	Delta
Total	9.786	in	time	0.002	in
Time	12.951	Total	delay	0.206	Delay
for	16.165	Times,	for	0.393	time:
break-	19.350	per tk:	Major	0.587	0.187
points:	22.141		Breaks:	0.781	0.194
	>>Mean:	0.193	>>Mean:	0.194	0.194

Disk stats:

RPM: 300
RPS: 5
SPR: 0.200
(Seconds per rev.)

Revolutions per head settle delay:

0.002	0.011 = 0
0.206	1.028 = 1
0.393	1.965 = 2
0.587	2.936 = 3
0.781	3.907 = 4

Table 2: Disk I/O speeds, 16-cylinder reads with different head settle time delay parameters

is taken immediately after a seek operation, subsequent processing must wait a full rotation before continuing. No further degradation occurs with increased delays until the delay equals or exceeds the rotational period, at which time processing time increases again by the rotation rate. The longer the BIOS waits for head settling, the less time *WAIT_INT* waits for the rotation of the disk to complete.

The data in Table 1, page 45, is inconclusive, however, because there is no entry for 400+ milliseconds and beyond: The interval between 2 and 204 milliseconds alone is not enough to prove the theory, and the test at 575 milliseconds has increased the delay too much to verify the 200-millisecond period.

Table 2 shows the results of additional tests that confirmed the hypothesis. If you examine the table, you will see that the jump from 12.9 to 16.1 seconds of processing time occurs at just about 400 milliseconds and from 16.1 to 19.3 at 600 milliseconds. Final confirmation is given by the jump at 781 (close enough to 800) milliseconds. Intervening data points in the 600- to 800-millisecond range are omitted, but timings showed the same plateaus as the other data.

Now we can see why DOS "can reach and sustain an input rate of one track per disk rotation." The trick is that by skipping the head settle delay altogether, there is no wait for an additional rotation of the disk and no latency degradation. One millisecond of delay is enough to force the controller to wait for another 200 milliseconds before the operation can complete.

Other Points

Returning to other points in Cortesi's column, it was simple to check more thoroughly into the problem of BASIC and disk I/O as outlined in the article. I found that indeed, as Cortesi writes (it's even documented in the manual), BASIC uses a default disk buffer size of 128 bytes. Increasing the buffer size can be done with a command-line switch: To specify a 1,024-byte disk buffer, the command is BASIC./S:1024.

I tried the test program with a buffer size of 8,192 bytes, and BASIC took only 10+ seconds to perform its DOS functions, as opposed to 44 sec-

PCTEX Now for PC Users: Professional Typesetting Capability

PCTEX brings to the personal computer user the ability to put any kind of information on paper in a professional, elegant manner. It brings the full power and flexibility of TeX implementations on mainframes to owners of IBM PC's, AT's and workalike computers. PCTEX is widely used for formatting technical and mathematical material. It is also perfectly suited for producing professional-quality reports, manuals, even books.

PCTEX offers a wide range of typefaces, and a wide choice of drivers which output the finished material on dot matrix printers (Epson, Toshiba), low-cost laser printers (Apple LaserWriter, Corona LP-300, HP Laser Jet) and graphics screen preview (Hercules, EGA). This ad was formatted by PCTEX and produced on a Corona LP-300.

Join hundreds of satisfied PCTEX users. Write or call us today.

PCTEX: only \$279. Dot-matrix drivers: \$100. Laser drivers: \$300. Preview (Hercules GC): \$250. MF Medley (44 fonts, including Computer Helvetica): \$100. Corona Laser Printer and PCTEX: complete \$3395. System requirements: DOS 2.0 or later, 512K RAM, 10M hard disk. M/C, Visa accepted.

Personal
TEX
Inc

20 Sunnyside, Suite H
Mill Valley, CA 94941
(415) 388-8853 Telex 275611

Trademarks: PCTEX, Personal TeX, Inc.; TeX, American Mathematical Society; IBM PC and AT, IBM Corp; LaserWriter, Apple Computer, Inc.; Hercules Graphics Card, Hercules Computer Technology.

Circle no. 76 on reader service card.

ALCYON:

*MC68K Q-BUS PROCESSORS
ALSO AVAILABLE*

BREAKING GROUND IN VME PROCESSING

- Announcing a high-density, high-performance MC68010 VMEbus processor board—The A68VME
- Includes on board RAM, ROM, extensive I/O including SCSI—a complete system in one VME slot
- Board resident applications module—add more I/O without additional boards or slots
- Full Alcyon software support—REGULUS, a Real-Time UNIX operating system and C68 optimizing C compiler



5010 Shoreham Place
San Diego, CA 92122
(619) 587-1155 TWX: 5106004947

UNIX is a trademark of AT&T REGULUS is a trademark of Alcyon Corp. Q-BUS is a trademark of Digital Equipment Corp.

Circle no. 221 on reader service card.

Make your PC or AT into a COMMUNICATING WORKSTATION for only \$85

Use ZAP, the Communications System for Technical Users
COMPLETE Communications for PROGRAMMING and ENGINEERING

EMULATION of graphics and smart terminals is combined with the ability to TRANSFER files reliably, CAPTURE interactive sessions, and transmit MESSAGES while also being able to swap between your mini or mainframe session and your PC application. SUSPEND a line to run a PC application. Reconfigure features to fit the communications parameters and keyboard requirements of the host computer software. Complete technical documentation helps you understand and fit ZAP to your style.

HIGHLIGHTS OF ZAP:

- Emulate TEKtronix 4010/14 and DEC VT 100, 102, 52 including variable rows and columns, windows, full graphics, even half tones.
- Reliable file transfer to/from any mainframes and PCs including KERMIT and XMODEM protocols plus you get a full copy of KERMIT. Transfer speeds ranging from 50 to 38,400 BAUD. Session control include *printer dumps* and *save to disk*.
- **MACRO and Installation files** ("scripts") controllable by you.
- EMACS, EDT and VI "Script" files are included. ZAP is also used with other popular software including graphics products like DISSPLA and SAS/GRAF.
- **CONFIGURABLE** to communications, terminal features on the "other end"; 1, 2 stop bits; 5, 6, 7 or 8 data bits; parity of odd, even, none, mark and space; remap all keys including the numeric pad and standard keyboard, set any "virtual" screen size.
- Full PC/MSDOS access to run any command or program that will fit in your systems memory. ZAP takes less than 64K.
- 9 Comm ports are supported by ZAP. Plus full color in text and graphics make use of the IBM color, EGA cards, or Hercules Monochrome.

ONLY
~~\$85~~

Full refund if not satisfied
during first 30 days.

Solution
Systems™

335-D Washington St.
Norwell, Mass. 02061
617-659-1571
800-821-2492

Circle no. 152 on reader service card.

Scrap your LINKER with FASTER C

Reliably:

CUT Compile times (by 15% to 55%)
CUT Testing times (by 12% to 37%)

HOW: FASTER C keeps the Lattice C or C86 library and any other functions you choose in memory. It manages a jump table to replace the LINKER and immediately execute your functions. You can also CALL active functions interactively to speed your program debugging. It includes many options for configuration and control.

"Automatic" support for new libraries by reading the .OBJ files makes support for new libraries quick and simple.

AVAILABLE FOR PC-DOS, IBM-AT,
AND ANY 256K MSDOS SYSTEM.

ONLY \$95.
Full Refund if not satisfied
during first 30 days.
Call 800-821-2492

Solution
Systems™

335-D Washington St., Norwell, Mass. 02061
617-659-1571

Circle no. 153 on reader service card.

SPEEDING MS DOS
(Continued from page 47)

onds with a 128-byte buffer. Because BASIC uses DOS for its file I/O, forcing the head settle time to 0 had no effect: The minimum value is in effect whenever DOS performs BIOS disk read calls.

All these results can be summarized as follows. First, there is no mystery, magic, or extraordinary cleverness involved in how DOS handles disk I/O, and it certainly uses the ROM BIOS. The magic goes away when all the facts are in, although there are new problems to solve.

According to my findings, when performing file I/O through DOS, you can expect the best performance when you read a block size of more than C000h bytes. DOS reads blocks almost an entire segment (64K) at a time when you use the COPY command. Why settle for less (no pun intended) if you have the memory?

When writing your own BIOS disk handlers using interrupt 13h, set the head settle time to 0 for all read operations. It must be longer for write operations, but you can ignore the overall impact the settling time will have on performance because you will catch the latency delay of 200 milliseconds. Use the documented required values for write head settling times. This has no effect when file I/O is performed through DOS because DOS takes care of the settling time parameter already.

You can, however, shave additional seconds from both your own disk I/O and DOS's with the motor-start-delay parameter: Reduce this value to the minimum value that does not cause errors when you first access the disk. (I found no problems with a start-up delay of 0.) Listing Three, page 100, shows you how to set the head settle time and motor start-up delay in an assembly-language program.

DDJ

(Listing begins on page 94.)

Reader Ballot
Vote for your favorite feature/article.
Circle Reader Service No. 5.

**Product
Information**

Free!

**Product
Information**

Postage Paid!

Subscribe And Save!

**Subscribe to
Dr. Dobb's Journal
and save over \$5—
a 15% savings off the cover price!**

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Please charge my: Visa MasterCard American Express
 Payment enclosed Bill me later

*For subscriptions outside the U.S. add \$27.00 airmail or
\$17.00 surface. All foreign subscriptions must be paid in U.S.
funds drawn on a U.S. bank.

This offer good until June 30, 1986. Please allow up to six
weeks for first issue.

A publication of M&T Publishing, Inc.

323S

\$35.40

\$29.97

**Subscribe
And
Save!**

**Subscribe to
Dr. Dobb's Journal
and save over \$5—
a 15% savings off the cover price!**

Card # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Please charge my: Visa MasterCard American Express
 Payment enclosed Bill me later

*For subscriptions outside the U.S. add \$27.00 airmail or
\$17.00 surface. All foreign subscriptions must be paid in U.S.
funds drawn on a U.S. bank.

This offer good until June 30, 1986. Please allow up to six
weeks for first issue.

A publication of M&T Publishing, Inc.

\$35.40

\$29.97

March 1986 #113

Dear Reader,

Dr. Dobb's has a long tradition of listening to its readers. We like to hear when something really helps or, for that matter, bothers you. In this hectic world of ours, however, it is often difficult to take the time to write a letter. This card provides you with a quick and easy way to correspond and, if you include your name and address, we may use appropriate comments in The Letters column. Simply fill it out and drop it in the mail.

—Ed.

Which articles or departments did you enjoy the most this month? Why?

Comments or suggestions:

Name _____

Address _____

Product Information

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 756 MENLO PARK, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of Software Tools
P.O. BOX 27809
SAN DIEGO, CA 92128

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 756 MENLO PARK, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of Software Tools
P.O. BOX 27809
SAN DIEGO, CA 92128

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

Free!

Postage Paid!

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 790 REDWOOD CITY, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of Software Tools
501 GALVESTON DR.
REDWOOD CITY, CA 94063

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

Product Information

Free!

Image-Pro™

The first device independent, easy to use, microcomputer based image capture and processing system that puts image processing tools into the hands of the professionals who need it.

With Image-Pro, image processing technology that was formerly available only on mainframes is now available on IBM AT's and compatibles. Whatever your application, Image-Pro fulfills your requirements for analyzing and processing images.

Image-Pro offers affordable, interactive, and user friendly solutions for:

- Image Analysis
- Image Enhancement
- Spatial Filtering
- Area Measurement
- Digitization, Storage and Retrieval
- Image Editing
- Half Tone Printing
- Report Generation and Annotation

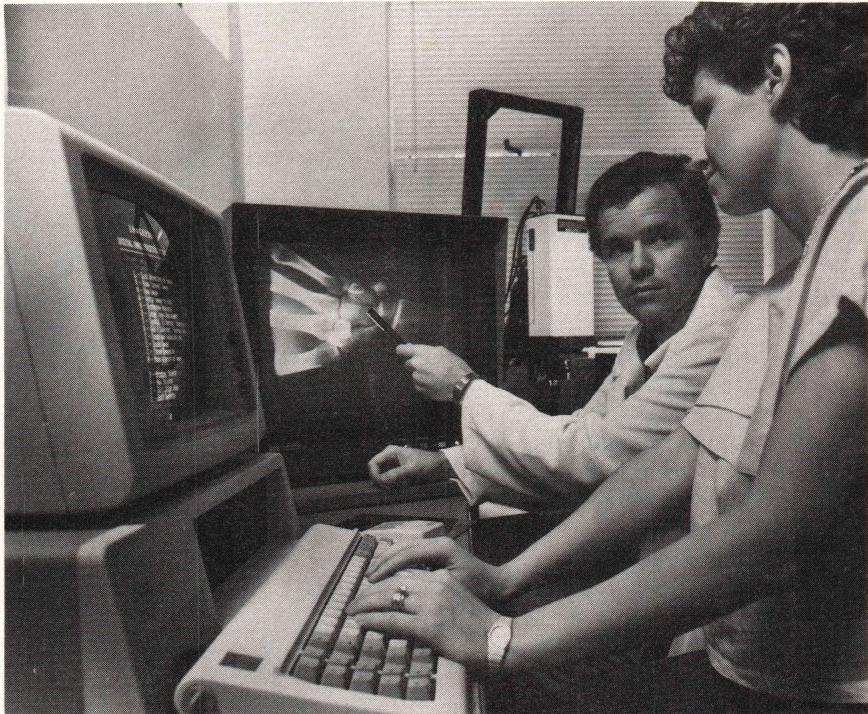
The heart of the solution is the Image-Pro Software. This software includes:

1. A stand alone interactive, image processing program.
2. An icon-based, image editor.
3. A library of powerful image processing subroutines.
4. A slide show and batch printing program.

With Image-Pro, information from documents, photographs, x-rays, satellites, etc., can be electronically captured, processed, analyzed, stored and retrieved for future reference.

Image-Pro Workstation

The Image-Pro Workstation includes: an IBM AT or compatible; precision video camera, image capture board, video display device, high resolution monitors, and a color or black and white printer (including laser printers).



And the Image-Pro software is transportable between Workstations configured with different imaging and graphics components.

Image-Pro Subroutines

Image-Pro subroutines provide software developers and OEMs with all the tools needed to quickly implement customized imaging solutions.

HALO Software Development Environment

With HALO, a complete programming environment is available as a com-

plementary package to Image-Pro. HALO is a powerful toolbox of over 170 graphics primitives that is an established standard for graphics in the IBM PC and compatibles marketplace.

Some of HALO's extensive functions include: point, lines, arc, circle, ellipse, image compression, rubberbanding, polygon fill, animation, windowing, color management, curve fitting, world coordinates, etc.

HALO supports the most popular programming languages, graphics controller cards, and input and output devices and includes LEARNHALO, an interactive, computer-aided tutorial.

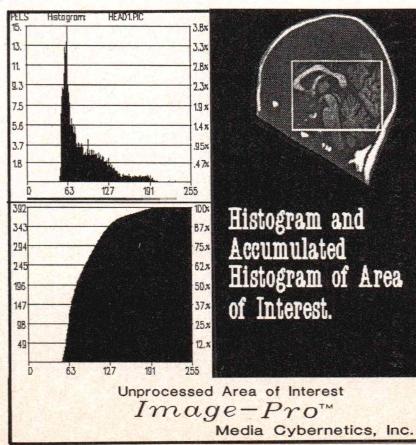
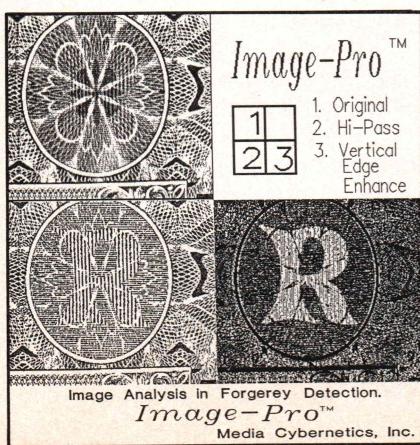
Ordering Information

The price for Image-Pro Software is \$1,000.00 for the standard software and \$2,000.00 for Image-Pro 1K. HALO costs \$250.00 for a single language; additional language bindings cost \$150.00.

For price information on the Image-Pro Workstation see GSA Schedule GS00K85AGS6100.

media cybernetics, inc.

8484 Georgia Avenue, Suite 200
Silver Spring, Maryland 20910
(301) 495-3305
(800) 446-HALO
Telex 322014



Automatic Porting Between Pascal Dialects

by Michael J. Sorens

DIALATE converts one dialect of Pascal to another, provided programs are written according to the guidelines.

```
function copy(STR: string255;
  WHERE, LEN: integer): string255;
begin
  copy := substr(str, where, len);
end;
```

Thereafter, we can use the *copy* function in either Turbo Pascal or VAX Pascal. Of course, we want the above definition of *copy* to exist only in the VAX environment because it will generate compilation errors in Turbo; we will see how to selectively compile this kind of entity shortly.

A second case might be this: In evaluating a logical expression, HP Pascal, for example, has a compiler option to do only partial evaluation (à la C) in which an expression is evaluated only until the point at which its result is determinable. C programmers use this to establish and avoid side effects. (It is quite a useful thing to have, but, alas, having it in only one Pascal dialect makes life difficult.)

If the function *foo* changes some global variable *y*, for example, then (*false and foo(x)*) will not change *y* if the dialect uses partial evaluation. This is because of the way the Boolean AND operates. Both terms must be true for the conjunction to be true. Scanning from left to right, because the first term *false* is false, we do not need even to look at *foo(x)* to deter-

In the course of programming events, some people have found it necessary to write Pascal code that could be ported between dialects of Pascal that differ from machine to machine. This task cannot be accomplished just by writing "vanilla" code to a maximal degree, as there are inevitably syntactical and semantic differences between dialects. At QCAD, we have developed a technique that, in conjunction with vanilla code, allows translation from one dialect to another to be performed automatically. Generally, when a file of program text needs to be translated to a new dialect, it also needs to be transmitted to a physically separate computer, and so our translation utility exists in two forms: a stand-alone translation utility and a combination translation/transmission utility.

Finding the Denominator

The primary task is to reduce functions and procedures to common denominators—that is, if a function does not exist in one dialect, it must be written for that dialect using the dialect's own primitives. This could be a simple renaming. Consider the following two functions, for example:

Turbo Pascal:

```
copy(string, location, length)
```

VAX Pascal:

```
substr(string, location, length)
```

As it happens, these functions are semantically identical, only the names have been changed to confuse the unwary. If we have written some code in Turbo Pascal, then we simply create a *copy* function for VAX Pascal:

© 1985 QCAD Systems Inc., 1164 Hyde Ave., San Jose, CA 95129

mine the result. Without the partial evaluation feature, *foo(x)* will always be evaluated, potentially yielding differing results.

A second example might be a test such as

```
while (count <= length(str)) and
  (str[count] = 'X') do ...
```

which is a valid statement with partial evaluation but can cause a run-time error without it. Why? Well, as long as *count* is less than or equal to the length of the string *str*, there is no problem. When *count* becomes one larger than the length of *str*, however, referencing *str[count]* may cause a run-time error, depending on whether range checking is enabled for a particular compiler.

Therefore, we must modify the code in which results might vary depending on whether partial evaluation is available. This could be done by introducing a two-step evaluation. That is, rather than *if (false and foo(x)) then ...*, we substitute *if false then if foo(x) then ...*, which guarantees an equivalent partial evaluation. The *while* loop requires introducing some ugliness. We need a Boolean variable—call it *done*—which we initialize to false. Then, the code might be

```
while (count <= length(str)) and
  (not done) do begin
  done := (str[count] = 'X');
  if not done then ...
```

The simple elegance and power of partial evaluation begins to shine through, no?

A more cumbersome reduction involves string comparisons. In HP Pascal or Turbo Pascal, you could compare strings *s1* and *s2* merely by interposing a relational operator—

for example, $(s1 < s2)$ or $(s1 = s2)$. In VAX Pascal, however, only strings of the same length can be compared (not strings of the same maximal length but strings of the same length at the moment of comparison). Thus, we need to introduce an added functional level in all three Pascals so that the code can be identical. We create the function *StrCmp* (see Table 1, below). Then, if we have existing code that needs to be retrofitted, we must change occurrences of $(s1 = s2)$ to $(strcmp(s1, s2) = 0)$ and likewise for the other relational operators.

Make That Code Disappear

At some point, there will be pieces of code that must be seen by one compiler but must be hidden from another. Enter DIALATE. *DIALATE*—a combination of the words *dialect* and *translate*—converts one dialect of Pascal to another, provided a program text file has been set up according to the guidelines about to be discussed.

We introduce a metanotation to be used in any Pascal we are interested in. This is just a notation that is in some sense “above” the Pascal code in that our translator will be looking only at the metanotation and not at the Pascal text itself. In our metanotation, we have metabrackets, which are the only constructs that DIALATE looks for:

```
{ @x } opening metabracket for
      dialect x
{ @ } closing metabracket
```

We use the curly braces { and } as the basis of our dialect notation because they already have the capability of hiding text from a Pascal compiler—the simple comment. DIALATE scans for comments that immediately begin with an @ symbol, indicating that the comment is a special, dialectic comment. Immediately following the @ can be one or more dialect designations. We currently use the following conventions:

```
T—Turbo Pascal (IBM PC)
V—VAX Pascal (VMS)
H—HP Pascal
A—Apple Pascal (Macintosh)
```

Any piece of code that cannot run on all relevant machines must be surrounded by metabrackets. DIALATE

inserts and removes the right curly brace of the opening metabracket in a judicious manner so that the compiler “sees” only valid language constructs.

Let's look at an example. Consider opening a file for input in Turbo Pascal and in HP Pascal:

```
Turbo Pascal:
  assign(MyFile, FileName);
  reset(MyFile);

HP Pascal:
  reset(MyFile, FileName);
```

Because there are significant differences, it is perhaps wise to create a procedure *OpenInputFile* that can be used in both Pascal dialects. Here is what the procedure will look like in Turbo Pascal:

```
(1)  procedure OpenInputFile
      (var F: text;
       NAME: string80);
(2)  begin
(3)    { @T }
(4)    assign(f, name);
(5)    reset(f );
(6)    { @ }
(7)    { @H }
(8)    reset(f, name);
(9)    { @ }
(10)   end;
```

Examine the position of each of the curly braces carefully. Notice that in line 3 there is a } but that in line 7 there is not. Thus, lines 4 and 5 are *active code*, while line 8 is *passive code*. Lines 6 and 9 are closing metabrackets that never change. The { in line 6 begins a comment that hides the @ character, while the { in line 9 is ignored because it is already inside

a comment. Now let's run the procedure through DIALATE, converting it to HP Pascal code:

```
(1)  procedure OpenInputFile
      (var F: text;
       NAME: string80);
(2)  begin
(3)    { @T
(4)    assign(f, name);
(5)    reset(f );
(6)    { @ }
(7)    { @H
(8)    reset(f, name);
(9)    { @ }
(10)   end;
```

The only difference is that the } in line 3 has gone, and there is a new } in line 7. This has reversed the active and passive sections of code. Hence, to write a piece of automatically translatable code, decide which dialect you wish to write in and use the appropriate metabrackets.

The technique is extensible to multiple machines with a concise notation. Take, for example, a function to find the location of a pattern string within some other string. In Turbo Pascal and HP Pascal, this function is called *pos*, while in VAX Pascal it is called *index*. We could then write a compatibility function called *index* to be used in Turbo or HP Pascal or one called *pos* to be used in VAX Pascal. A third alternative, though, is to write a function with a new name, perhaps *LocateSubString*, to be used in all three languages. Stylistically, it might be better to use a very different name so that there is no chance of confusing the name with some other valid language construct. As it hap-

```
function StrCmp(S, T: string80): integer;
{ return -1 if s < t; 0 if s = t; 1 if s > t }
var Result: integer;
begin
  if length(s) < length(t) then begin
    result := StrCmp(s, copy(t, 1, length(s)));
    if result = 0 then StrCmp := -1 else StrCmp := result
  end
  else if length(s) > length(t) then begin
    result := StrCmp(copy(s, 1, length(t)), t);
    if result = 0 then StrCmp := 1 else StrCmp := result
  end
  else if s = t then StrCmp := 0
  else if s < t then StrCmp := -1
  else if s > t then StrCmp := 1
end;
```

Table 1

PASCAL PORTING

(Continued from page 51)

pens, the functions *pos* and *index* do precisely the same thing, though their parameter order is different, so we do not have to write much code.

```
(1) function LocateSubString
    (Object,
     Target: string80): integer;
(2) begin
(3)   { @HT
(4)   LocateSubString := 
        pos(object, target);
```

```
(5)   { @ }
(6)   { @V }
(7)   LocateSubString := 
        index(target, object);
(8)   { @ }
(9) end;
```

We can see that the above function is written in VAX Pascal because line 7, the VAX code, is active. Line 4 is passive code that is for both the HP and the Turbo dialects because line 3 has both a *T* and an *H*. When translated to either of these dialects, line 4 becomes active, while line 7 becomes passive.

Case in Point

One final twist has precipitated out of the Babel-like differences in Pascals, and that is the *else* clause of a *case* statement. Some Pascals, such as Turbo, use the keyword *else* to indicate any cases not explicitly enumerated. Other Pascals, such as VAX, HP, and Macintosh, use the keyword *otherwise*. We can certainly handle this discrepancy using our standard metanotation described in the previous section. A typical *case* statement might look like this:

```
case SelectionChar of
  'R': RunIt;
  'P': ProcessIt;
  'E': EditIt;
  { @T } else { @ }
  { @HVA otherwise { @ }
    writeln('Invalid selection
            character);
end { case };
```

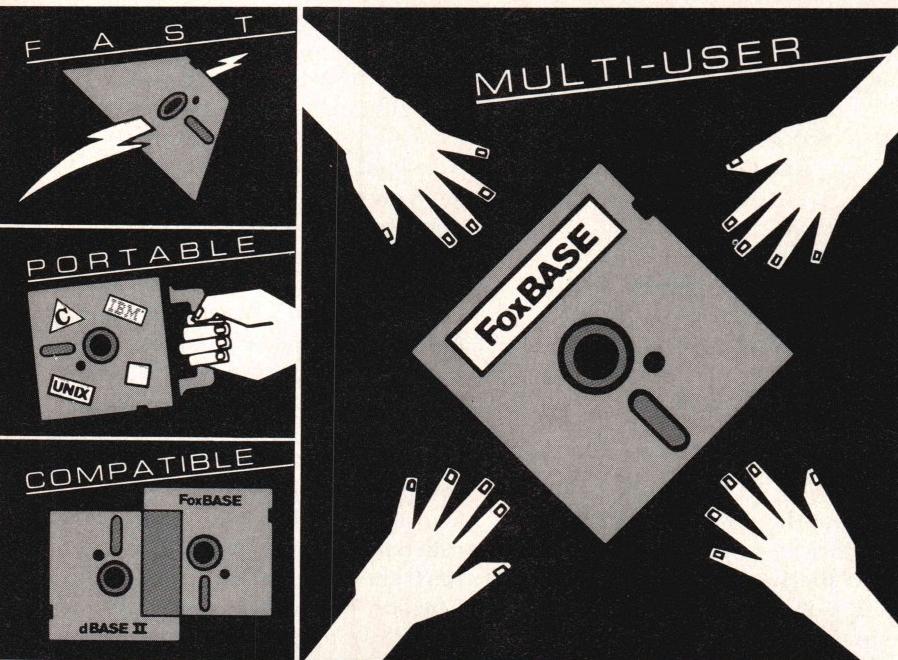
This can quickly become an annoyance, though. Because there is no way to make some kind of generic function that handles all *case* statements (as we did with *OpenInputFile*), we must use the metabrackets every time we have a *case* statement. Or rather, we would have to if *DIALATE* didn't have a better solution.

What we would like to do is have the translator automatically convert an *else* to an *otherwise* if we are going from Turbo to either HP, VAX, or Macintosh Pascal and convert an *otherwise* to an *else* if we are going in the converse direction. But wait, what about the oft-found *if...then...else...* statement? How will we know if an *else* belongs to an *if* or to a *case*?

What we have chosen to do is have a special ELSE-OTHERWISE convention. In Turbo Pascal, we write *ELSE* to denote an *else* of a *case* statement and *else* to denote an *else* of an *if* statement. In our other Pascals, we write *OTHERWISE* to denote an *otherwise* of a *case* statement and *else* to denote an *else* of an *if* statement. That is, the *case* clause—whatever it is called—must be in uppercase letters, while the *if* clause must be in lowercase letters.

Discussion

This dialect translation technique has minor disadvantages. The foremost is



FoxBASE.
The DBMS That
Bridges The Gap
Between Single-user
And Multi-user.

Unsurpassed Program Development Speed.

FoxBASE™ uses a state-of-the-art B+ Tree index structure for quicker, more efficient data access. A sophisticated virtual storage technique becomes an invaluable timesaving device as it works to insure that frequently referenced programs are retained in memory in compiled form. What's more, FoxBASE provides automatic 8087/80287 math coprocessor support for ultraquick program execution speed—as much as six times the speed of dBASE II!™

Highly Portable.

FoxBASE is much more than a relational database management system. Written in C, FoxBASE is an extremely portable interpreter/compiler. Now you can port from one machine or operating system to another without changing your applications. And this portability protects your investment in programs by insuring their use in future machine and operating system environments.

dBASE II Compatible.

FoxBASE is both source language—including full macro usage—and data file compatible with the dBASE II database language. This means your existing dBASE II databases can be used unchanged. Furthermore, it puts thousands of public-domain and commercially available dBASE II programs at your disposal.

Available Under Multi-user Systems.

Our multi-user versions of FoxBASE feature many additional enhancements. Like automatic file-locking and record-locking capabilities. Use of termcap, so FoxBASE can run on virtually any terminal. And, with some versions, a two billion record file capacity.

Multi-user Versions:

Xenix™ \$995. MultiLink™ \$995.

IBM-PC/PC-NET™ \$995.

Single-user Versions:

MS/PC-DOS™ \$395. AOS/VS \$995.

**Don't be outfoxed by the others.
Call or write Fox Software today.**

FOXBASE™
From
FOX SOFTWARE, INC.

27475 Holiday Lane, Perrysburg, OH 43551
419-874-0162

Circle no. 94 on reader service card.

that you must have all metabrackets balanced and correct, otherwise you might hide too little or too much code from the compiler. This might cause compilation errors, but it also might not, possibly creating subtle bugs. If we accidentally made the VAX code above into passive code, for example, then the *LocateSubString* function would never be assigned a value. This could cause random or unpredictable results in the program. On the other hand, if we made both assignments active, a compiler should complain about the unknown function *index* or *pos*, depending on which machine the code is compiled. It takes a little getting used to, but typing correct metabrackets is, after all, no more difficult than typing correct syntax in a programming language.

Second, it is not possible to put actual comments inside a region that is metabracketed. This may cause compilation errors when the entire region is supposed to be hidden because the closing comment bracket could inadvertently reactivate a portion of the hidden code. This is most insidious, however, when it does not cause compilation errors, as, for example:

```
procedure DUMMY;
begin
  { @H
  a := 10;
  b := 5;
  { @ }
  { @T }
  a := 5; { some global variables }
  b := 10;
  { @ }
end;
```

The above code, as written, will work fine with the Turbo compiler. When we translate it to run with HP Pascal, however, we will get the wrong value for *b* because the closing } of the "real" comment will prematurely close the comment created by the metabrackets.

The dialect translation technique discussed in this article is an effective and rapid way to work in several different Pascals with virtually the same program text. It may seem awkward at first, but you can readily get used to the style. And for those who need to work with more than one dialect or more than one machine, the tool may prove invaluable.

DIALATE was developed because of a perceived need at QCAD. We manufacture a large software package (a parser generator) that runs on the machines discussed above; DIALATE allows us to keep the same code on all the machines.

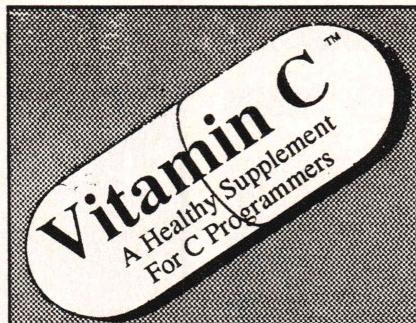
Should there be any readers who are so amazed by the technique revealed in this article (for the first time anywhere), I will gladly supply

both object and source code for DIALATE for a mere \$10.24 (a kilopenny) to cover diskette, postage, copying, and so on. I will also be happy to tell you about some of the neat software tools that QCAD sells for real money.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 6.



Perfect Windows + Powerful Data Entry already integrated!

Vitamin C's versatile features include...

- Complete input formatting
- Unlimited validation
- Full attribute control
- Field sensitive application help system
- Multiple virtual windows
- Fully automatic, collision proof overlay and restore
- Print to & scroll background windows
- Animated window "zoom"
- Move, grow, shrink, hide, or show any window
- Date & time arithmetic routines
- "Loop function" allows processing while awaiting input

...and much much more!

Finally! A library of high level C functions (not just a bunch of building blocks) designed to increase your productivity and help develop superior applications in dramatically less time! How? Well, Vitamin C automatically coordinates the complex tasks and leaves the programmer free to be creative! With Vitamin C, for example, you'll never even have to think about saving or restoring portions of the screen when a window is opened, closed or moved. Simply call wopen(), wclose() or wmove() and Vitamin C takes care of the complexities! It's just that easy! This philosophy of relieving the programmer from as many details as possible runs throughout Vitamin C. As a result, jobs that used to take days are finished in a matter of hours!

Includes 100% source, reference manual, step by step tutorial, examples, sample programs. Specify Microsoft v3, Lattice, Aztec, Computer Innovations, DeSmet or Mark Williams. Ask about UNIX, TI-Pro and other compatibility!

Vitamin C \$149.95

100 % MONEY BACK GUARANTEE

Better than a brochure. More informative than a testimonial. Our guarantee gives you the opportunity to see first hand why programmers who demand performance are using Vitamin C.

Find out for yourself why Vitamin C users are saying...

The best structured and documented C source package we have ever seen.

Thanks to Vitamin C, our projects are back on schedule.

I own them all, but I USE Vitamin C!

NEW! VC Screen NEW!

Our new interactive screen "painter" actually lets you draw your input screens. Define fields, text, boxes & borders. Move them around. Change their attributes. Then after everything is just the way you want it, the touch of a button generates C source code calls to Vitamin C routines OR generates parameter files that will dynamically load & build each screen at run time. Either way, your screen designs will be faster & more pleasing with VC Screen! Requires Vitamin C Library.

VC Screen \$99.95

Available for the IBM PC/XT/AT and compatibles.

For Orders Or More Information...

(214)245-6090

Creative Programming
Consultants
Box 112097

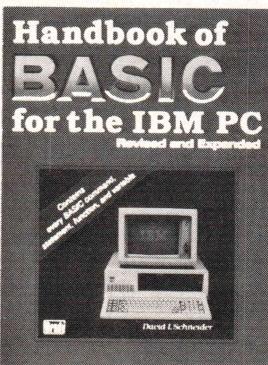
Carrollton, Texas 75011-2097

Please add \$3 ground, \$6 air, \$15 overnight shipping per unit inside USA, \$25 per unit if outside USA. Texas residents add 6 1/8% sales tax. All funds must be in U.S. dollars drawn on a U.S. bank.

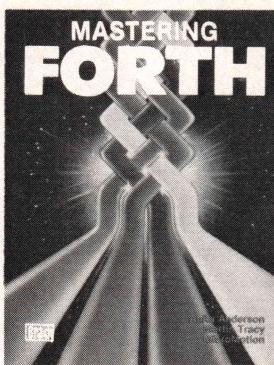
Circle no. 82 on reader service card.

BRADY Speaks Your Language

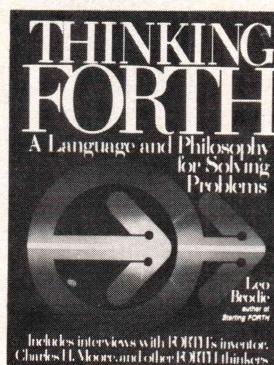
We can help you buff up your BASIC...Put a finish on your FORTH...Conquer C...Succeed with Assembler...And more! Just call toll-free or use the coupon to order today!



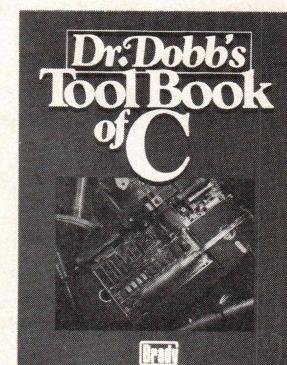
1. PC magazine calls it: "A truly remarkable book... A treasure trove of useful programming information for the IBM PC." \$22.95



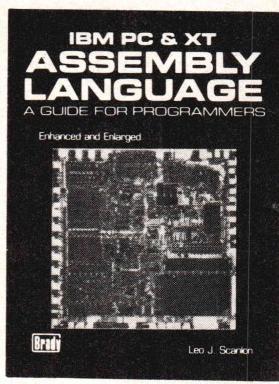
2. A step-by-step tutorial for the high-level, stack-oriented FORTH-83 standard. \$17.95



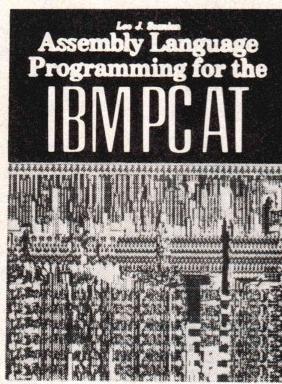
Includes interviews with RUMI's inventor Charles H. Moore and other RUMI thinkers.



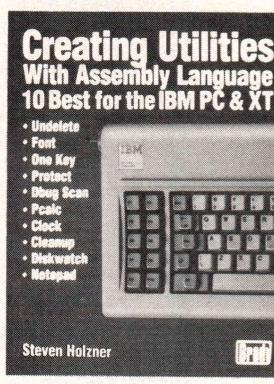
4. The best C articles from the highly respected Dr. Dobb's Journal dealing exclusively with C language and programming techniques. \$24.95



5. Our best-selling assembler book has been made even better! It now includes 30 assembler Macros and version 2.0 of the IBM Assembler. \$21.95



6. The author of our best-selling assembler books now demonstrates his detailed and accurate style on the 80286 chip. \$21.95



7. For the more advanced user familiar with Assembly language, here's an opportunity to unleash the power of 10 DOS-enhancing programs. \$21.95



8. Probes the inner workings of the 8086 (used by the AT&T 6300) and 8088 (IBM PC) chips...and describes specific techniques for using the full capability of these chip designs while programming in assembler. \$18.95

Now at your book or computer store.
Or order toll-free today:

800-624-0023

In New Jersey:
800-624-0024

Brady COMMUNICATIONS COMPANY, INC.
c/o Prentice Hall,
P.O. Box 512, W. Nyack, NY 10594

Circle the numbers of the titles you want below. (Payment must be enclosed; or, use your charge card.) Add \$1.50 for postage and handling. Enclosed is check for \$_____ or charge to MasterCard VISA.

1 (0-89303-510-6)
5 (0-89303-575-0)

2 (0-89303-660-9)
6 (0-89303-484-3)

3 (0-13-917568-7)
7 (0-89303-584-X)

4 (0-89303-599-8)
8 (0-89303-424-X)

Acc't # _____ Exp. date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____
(New Jersey residents, please add applicable sales tax.)
Dept. 3

GLDDJ-AO(9)

LETTERS

LISTING ONE (Text begins on page 8)

```
*****
/*
 *      Bose-Nelson Sort - Recursive Version
 *
 *      by R J Wissbaum
 *      15553 E Wyoming Dr - H
 *      Aurora, CO    80012
 *
 *      Reference:
 *          Dr. Dobb's Journal
 *          September 1985 (#107)
 *          Page 68
 *
 *      #include "STDIO.H"
 *
int count = 0;
main ( argc, argv )
int argc, *argv;
{ int n;
    char arg[255];
    if ( argc < 2 )
        { printf ( "USAGE: bose5 n >outfile \n" );
        exit();
    }
    else
        { getarg( 1, arg, 255, argc, argv );
        n = atoi( arg );
        if ( n<0 ) n = -n;
        boosesort ( 2, 1, n, 0, 0 );
        printf ( "There were %d swaps \n", count );
    }
}
boosesort( n, i, x, j, y )
int n, i, x, j, y;
{ int a, b;
switch ( n )
{
case 0: { /* printf ( ")\n" ); */
}
break;

case 1: { /* printf ( "swap ( %d, %d ); \n", i, x ); */
count++;
}
break;

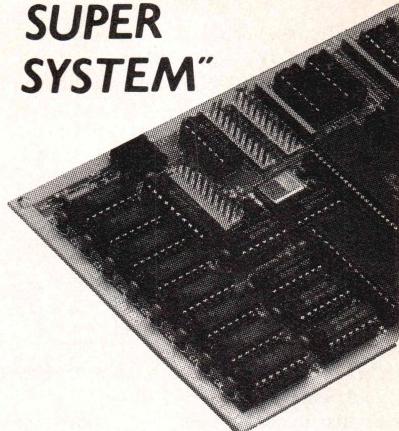
case 2: { if ( x != 1 )
    { a = ( x / 2 );
    boosesort ( 3, i, a, (i+a), (x-a) );
    boosesort ( 2, (i+a), (x-a), 0, 0 );
    boosesort ( 2, i, a, 0, 0 );
    }
}
break;

case 3: { a = ( x / 2 );
if ( even ( x ) ) b = ( y + 1 ) / 2;
else                b = ( y / 2 );
if ( ( x == 1 ) && ( y == 1 ) ) boosesort ( 1, i, j, 0, 0 );
else if ( ( x == 1 ) && ( y == 2 ) )
{
    boosesort ( 1, i, j, 0, 0 );
    boosesort ( 1, i, (j+1), 0, 0 );
}
else if ( ( x == 2 ) && ( y == 1 ) )
{
    boosesort ( 1, (i+1), j, 0, 0 );
    boosesort ( 1, i, j, 0, 0 );
}
else if ( ( x != 0 ) && ( y != 0 ) )
{
    boosesort ( 3, (i+a), (x-a), j, b );
    boosesort ( 3, (i+a), (x-a), (j+b), (y-b) );
    boosesort ( 3, i, a, j, b );
}
}
break;
default:
printf ( "FATAL: Error in stack \n" );
exit();
}
break;
} /* End of WHILE loop */
} /* End of BOSE SORT */
even ( x )
int x;
{ return ( 1 - ( 1 & x ) );
}
```

End Listing

Byte Magazine called it.

"CIARCA'S SUPER SYSTEM"



The SB180 Computer/Controller

Featured on the cover of Byte, Sept. 1985, the SB180 lets CP/M users upgrade to a fast, 4" x 7½" single board system.

- **6MHz 64180 CPU** (Z80 instruction superset), 256K RAM, 8K Monitor ROM with device test, disk format, read/write.
- **Mini/Micro Floppy Controller** (1-4 drives, Single/Double Density, 1-2 sided, 40/77/80 track 3½", 5¼" and 8" drives).
- **Measures 4" x 7½", with mounting holes**
- **One Centronics Printer Port**
- **Two RS232C Serial Ports** (75-19,200 baud with console port auto-baud rate select).
- **Power Supply Requirements** +5V +/-5% @500 mA +12V +/- 20% @40mA
- **ZCPR3 (CP/M 2.2/3 compatible)**
- **Multiple disk formats supported**
- **Menu-based system customization**

SB180-1

SB180 computer board w/256K bytes RAM and ROM monitor \$369.00

SB180-1-20

same as above w/ZCPR3, ZRDOS and BIOS source..... \$499.00

-Quantity discounts available-

NEW

COMM180-M-S

optional peripheral board adds 1200 bps modem and SCSI hard disk interface.

**TO ORDER
CALL TOLL FREE
1-800-635-3355**

**TELEX
643331**

For technical assistance or to request a data sheet, call:
1-203-871-6170



**Micromint, Inc.
25 Terrace Drive
Vernon, CT 06066**

Put More UNIX™ in Your C.

Unitools \$99

MAKE, DIFF and GREP



These versatile UNIX-style utilities put power at your fingertips. MAKE, a program administrative tool, is like having an assistant programmer at your side. DIFF compares files and shows you the differences between them. GREP can search one or many files looking for one pattern or a host of patterns.



"Z" \$99

A Powerful "vi"-type Editor:

Similar to the Berkeley "vi" editor, "Z's" commands are flexible, terse, and powerful; macro functions give you unlimited range. Features include "undo," sophisticated search and replace functions, automatic indentation, C-tags, and much, much more.



PC-LINT \$99

Error Checking Utility

A LINT-like utility that analyzes programs and uncovers bugs, quirks and inconsistencies. Detects subtle errors. Supports large and small memory models, has clear error messages and executes quickly. Has lots of options and features that you wouldn't expect at this low price.



SunScreen \$99

Low-priced Screen Utility

This versatile graphics package easily creates and modifies formatted screens, validates fields, supports function keys, color and monochrome cards. With library source SunScreen is \$199.

**Compatible with all leading MS/
PC-DOS C compilers.**

SPECIAL OFFER:

Unitools, "Z,"

PC-LINT and Sun-

Screen All for only

\$349

To order or for information call:

TECWARE
1 800 TEC WARE
(In NJ call 201-530-6307)



C CHEST

LISTING SIX (Text begins on page 14)

```

1 char *skipto( c, p, esc )
2 register char *p;
3 register int c, esc ;
4 {
5     /* Skip to c or to end of string. If c is preceded by
6      * the esc character it is skipped over.
7      */
8     while( *p && *p != c )
9     {
10         if ( *p != esc )
11             p++;
12         else if ( ++p ) /* skip over escaped characters */
13             p++;
14     }
15     return(p);
16 }
17
18
19 }
```

End Listing Six

LISTING SEVEN

```

1 /*
2  * DIR.H
3  *      #defines and typedefs needed to talk to the routine dir().
4  *      A pointer to the DIRECTORY structure is passed to dir().
5  *      DIRECTORY structures are created by mk_dir() and deleted with
6  *      del_dir().
7  *
8  *      On entry:
9  *          dirv    is the first of an uninitialized array of character
10 *                  pointers.
11 *          lastdir should be initialized to point at dirv.
12 *          maxdirs is the size of the above
13 *          nfiles
14 *          ndirs
15 *          nbytes is the total file count, the total directory count
16 *                  and the total byte count. These will be incremented
17 *                  as appropriate and are usually set to 0 before
18 *                  calling dir().
19 *          width   should be initialized to 0 before calling dir().
20 *          vol_label is undefined on entry to dir().
21 *          longf   is 1 if entries are to be printed in long format
22 *          files   is 1 if files are included in the list.
23 *          dirs    is 1 if directors are included in the list.
24 *          graphics is 1 if directories are highlighted with boldface.
25 *          hidden   is 1 if hidden files are to be included in the list.
26 *          path    is 1 if the path name is to be included in the list.
27 *          label   is 1 if you want to get the volume label
28 *          exp     is 1 if you want the contents of a subdirectory to
29 *                  be listed rather than the directory name when. This
30 *                  is only looked at if no wild cards are present in
31 *                  the file spec.
32 *          sort    is 1 if you want the list sorted.
33 *
34 *      all other fields are ignored. On exit the structure will have been
35 *      updated as follows:
36 *
37 *          lastdir will be incremented to point at the last entry added
38 *                  to the dirv table.
39 *          maxdirs will be decremented to reflect the added entries.
40 *          nfiles is the number of added entries which are files.
41 *          ndirs  is the number of added entries which are directories.
42 *                  Note that the equivalent of argc can be derived by
43 *                  adding ndirs and nfiles together.
44 *          nbytes will have the total size in bytes of all files added
45 *                  to dirv. This number is the number of bytes actually
46 *                  occupied by the file, ie. the size returned by DOS
47 *                  rounded up to the nearest multiple of the disk's
48 *                  cluster size.
49 *          vol_label will hold the volume label provided that "label" was
50 *                  set on entry.
51 *          width   will hold the length of the widest entry added to dirv
52 *
53 *      all other fields will have the same values they had on entry.
54 */
55
56 typedef struct
57 {
58     char    **lastdir; /* Most recent addition to dirv */
59     int     maxdirs; /* # of free slots in dirv */
60     int     nfiles; /* # of used slots that are files */
61     int     ndirs; /* # of used slots that are directories */
62     long    nbytes; /* byte count of files */
63     char    vol_label[12]; /* volume label if requested */
64 }
```

UNIX is a registered TM of Bell Laboratories. MANX AZTEC TM Manx Software Systems, Inc. PC
LINT TM GIMPLE software. SunScreen TM SunTec. MS-DOS TM Microsoft.

Circle no. 223 on reader service card.

```

64     unsigned width : 7; /* Width of widest element in dirv */
65
66     /* Various flags control how dir works: */
67     unsigned longf : 1; /* Use long format for entries */
68     unsigned files : 1; /* Include files in list */
69     unsigned dirs : 1; /* Include directories in list */
70     unsigned graphics : 1; /* Use graphics around directory names */
71     unsigned hidden : 1; /* List hidden files */
72     unsigned path : 1; /* List complete path if given */
73     unsigned label : 1; /* Load vol_label with volume label */
74     unsigned exp : 1; /* Expand sub-directories */
75     unsigned sort : 1; /* Sort added entries */
76
77     char dirv[1]; /* The first of the dirv entries */
78 }
79 DIRECTORY;

```

End Listing Seven

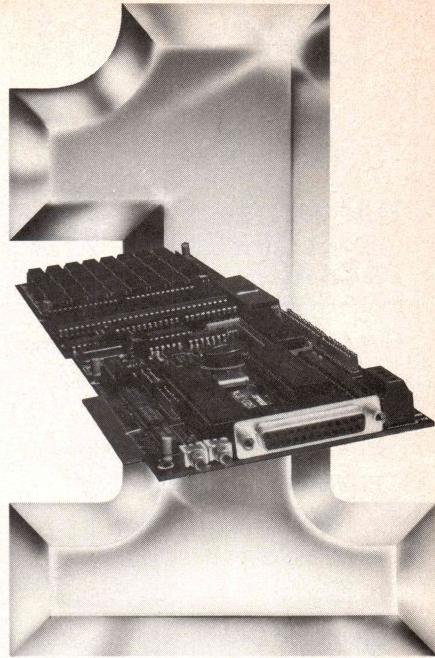
LISTING EIGHT

```

1 #include <stdio.h>
2 #include <getargs.h>
3 #include <mydos.h>
4 #include <dir.h>
5
6 /**
7 * DIR.C: An MSDOS directory access function.
8 *
9 * (c) Copyright 1985, Allen I. Holub. All rights reserved.
10 */
11 * 11/22/85 Modified so that the total amount of disk space used
12 * (ie. # of clusters) is put into the total, rather
13 * than the file size.
14 */
15 */
16
17 /* ROUND(n,u): if n is an even multiple of u, evaluate to n, else
18 * round n up to the next even multiple of u.
19 */
20
21 #define ROUND(n,u) ( !((n) % (u)) ? (n) : (((n) / (u)) + 1) * (u))
22
23
24 #define BOLDFACE "\033[1m" /* Ansi esc sequence to turn bold face on */
25 #define ALL_OFF "\033[0m" /* attributes off */
26
27 #define ATTRIBUTES (READONLY | DIRTY | SYSTEM | HIDDEN | SUBDIR)
28 #define iswhite(c) ((c) == ' ' || (c) == '\t')
29
30 */
31
32 extern char *calloc (unsigned,unsigned); /* In standard library */
33 extern char *cptolower(char*,char*); /* In /src/tools/cptolow.c */
34 extern char *cpy (char*,char*); /* In /src/tools/cpy.c */
35 extern int dos (REGS *); /* In /src/tools/dos.asm */
36 extern void gregs (REGS *); /* In /src/tools/dos.asm */
37 extern char *malloc (unsigned); /* In standard library */
38 extern char *next (char**,int,int); /* In /src/tools/next.c */
39 extern void ssort (char*,int,int,int(*)()); /* In /src/tools/ssort.c */
40 extern int strcmp (char*, char*); /* in standard library */
41
42 */
43
44 static unsigned Longfmt = 0; /* True if we're using long format. This
45 * has to be global for the comparison
46 * routine used for sorting to work.
47 */
48
49 static unsigned Cluster_size; /* Number of bytes per cluster on
50 * requested disk.
51 */
52
53 */
54 /* Do a DOS system call using the dos() routine */
55
56 #define DOSCALL(id,regs) { regs.h.ah = id ; dos( &regs ); }
57
58 */
59
60 static int find_first( filespec, attributes, regp )
61 char *filespec;
62 short attributes;
63 register REGS *regp;
64 {
65     /* Get directory information for the indicated file.
66     * Ambiguous file references are ok but you have to use
67     * find_next to get the rest of the file references.
68     * In this case, The regp structure used by find_first
69     * must be passed to find_next. 0 is returned on success,
70     * otherwise the DOS error code is returned.
71     */
72

```

(Continued on next page)



**Number One
in Performance**
68010/68000
Coprocessor for
IBM/AT/XT/PC-
8/10/12.5mz No Wait States
\$1295⁰⁰ Qty. 1

FEATURES

- 1-2 MB RAM (1MB Standard)
- 16K-64K EPROM
- 2-8 Serial Ports
- Async/Sync/Bisync Communications
- Battery-backed Real Time Clock
- Battery-backed 2K-8K RAM
- 2 Parallel Ports
- 68811 Math Coprocessor
- Memory-mapped Dual-port BUS
- 3-9 Users Per Board (3 Standard)
- Up To 16 Boards Per AT/XT/PC
- Can Operate As Standalone Processor

SOFTWARE

- OS9 (Powerful UNIX-like Multi-user OS)
- CPM/68K
- Software selectable OS including concurrent PC DOS/OS-9 or CPM/68K operation
- Support Module for IBM Graphics
- High-speed Local/Global Disk Caching
- Basic, Pascal, Fortran, C, and COBOL

IBM is a registered trademark of International Business Machines Corp. CPM is a registered trademark of Digital Research Corp. OS-9 is a registered trademark of Microsoft Systems Corp. CPM/68K is a registered trademark of Digital Research Corp. 68000/68010 is a registered trademark of Motorola. UNIX is a registered trademark of AT&T.



West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345
East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450
Distributor: Telemarketing Services, Inc.
1897 Garden Ave., Eugene, OR 97403, 800-874-2288

Circle no. 173 on reader service card.

DeSmet C

**MACINTOSH™
DEVELOPMENT
PACKAGE** \$150
Includes Shipping

Runs on 128K and 512K Macintosh

- Produces FINDER/SHELL applications
 - Dynamic OVERLAY support

Full K&R C Compiler

- Native Code Compiler
 - In-line **asm** directive
 - IEEE S/W Floating Point

Assembler, Linker, and Librarian

Machine Code Debugger

Source Code Editor

“SHELL” Interface

- Environmental Variables
 - Wild-Card Expansion
 - Many Built-in Functions
 - Command History
 - Runs Any Application

>120 Function STDIO Library

>450 Function Macintosh ROM Library

360 Page Manual

RAM Disk

Macintosh is a trademark licensed to Apple Computer, Inc.

- Please Send Information.
- Send Macintosh Development Package

Check # _____ Enclosed

SHIP TO:

ZIF

CWARE

CORPORATION

P.O. BOX C
Sunnyvale, CA 94087
(408) 720-9696

All orders shipped UPS surface. California residents add sales tax. Canada shipping add \$5, elsewhere add \$15. Checks must be on U.S. Bank and in U.S. Dollars.

Call 9 a.m.-1 p.m. to CHARGE by VISA/MC/AMEX

C CHEST

LISTING EIGHT (Listing continued, text begins on page 14)

```

73     regp->h.ah = (char) FINDFIRST ;
74     regp->x.dx = (short) filespec ;
75     regp->x.cx = attributes ;
76
77     return (int)( (dos(regp) & CARRY) ? regp->x.ax : 0 );
78 }
79
80 /*-----*/
81
82 static int      find_next ( regp )
83 REGS      *regp;
84 {
85     /* Get the next file in an ambiguous file reference. A
86      * call to this function must be preceded by a
87      * find_first call. The regp argument must be the
88      * same register image used by the find first call.
89      * 0 is returned on success, otherwise the error code
90      * generated by DOS is returned.
91     */
92
93     regp->h.ah = FINDNEXT ;
94     return (int)( (dos(regp) & CARRY) ? regp->x.ax : 0 );
95 }
96
97 /*-----*/
98
99 int      haswild(s)
100 register char   *s;
101 {
102     /* Return true if s has a unix wild card in it. */
103
104     for( ; *s ; s++)
105         if( *s == '*' || *s == '?' )
106             return 1;
107
108 }
109
110 /*-----*/
111
112 static int      isrootdir( name )
113 register char   *name;
114 {
115     /* return true if name is explicitly specifying the root
116      * directory (ie. is one of: d:/ d:\ / \ where
117      * 'd' can be any disk designator.
118     */
119
120     if( *name && name[1] == ':' )
121         name += 2;
122
123     return( (*name == '\\\\' || *name == '/') && !name[1] );
124 }
125
126 /*-----*/
127
128 hasonly( str, inclusion_set )
129 register char   *str;
130 char           *inclusion_set;
131 {
132     /* Return true only if every character in str is also in
133      * inclusion_set. True is returned if str is empty.
134     */
135
136     register char   *p;
137
138     for( ; *str ; str++)
139     {
140         for( p = inclusion_set ; *p && *p != *str ; p++ )
141             ;
142
143         if( !*p )
144             return 0;
145     }
146
147     return 1;
148 }
149
150 /*-----*/
151
152 static char      *fixup_name( name, regs, info )
153 register char   *name;
154 REGS      *regs;
155 FILE_INFO    *info;
156 {
157     /* If the name specifies an implicit file (ie. it asks for
158      * the directory rather than the files in the directory),
159      * modify it to ask for files (eg. ".." becomes "..\*.*").
160      * If the name is actually modified, a pointer to a modified
161      * copy of the original name is returned. Otherwise the
162      */

```

```

162     * original buffer is returned.
163     */
164
165     static char buf[80] ;           /* Place to put modified name */
166     register char *p = buf ;      /* Always points into buf */
167     char *start_name = name; /* Remember start of name */
168
169     if( isrootdir(name) || (name[0] && name[1]==':' && !name[2]) )
170     {
171         /* Handle an explicitly requested root directory or
172          * the current directory on another disk.
173          */
174
175         sprintf(buf, "%s.*", name );
176     }
177     else if( !find_first( name, ALL, regs) )
178     {
179         /* Look for the indicated name & see if it's a directory.
180          * If so, append slash-*.* to the requested name
181          */
182
183         if( !IS_SUBDIR(info) )
184             return name;
185         else
186             sprintf(buf, "%s/*.*", name );
187     }
188     else
189     {
190         /* If we get here then a non-existent file or directory
191          * was requested.
192          * If the name consists of nothing but the characters
193          * \ / . or a drive designator, assume that the root
194          * directory was requested and adjust the name
195          * accordingly.
196          */
197
198         if( *name && name[1] == ':' ) /* Copy drive designator if */
199         {
200             /* one's present. */
201             *p++ = *name++ ;
202             *p++ = *name++ ;
203         }
204
205         if( has_only(name, ".\\\"/) )
206             strcpy( p, "/*.*" );
207         else
208             return( start_name );
209     }
210
211     return( buf );
212 }
213 */
214
215 static int dirtoa( target, infop, graphics, pathname )
216 register char *target ;
217 char *pathname ;
218 register FILE_INFO *infop ;
219 unsigned graphics;
220 {
221     /* Convert directory entry held in infop to an ascii string
222      * in target. If Longfmt use a long format, if graphics then
223      * directory names are printed in bold face, else they're
224      * printed as "<name>." If pathname is true then the name
225      * will be preceded with the full pathname.
226      */
227
228     char *startstr = target;
229     int i;
230
231     if( Longfmt )
232     {
233         *target++ = ( IS_READONLY(infop) ) ? 'r' : '.';
234         *target++ = ( IS_HIDDEN (infop) ) ? 'h' : '.';
235         *target++ = ( IS_SYSTEM (infop) ) ? 's' : '.';
236         *target++ = ( IS_SUBDIR (infop) ) ? 'd' : '.';
237         *target++ = ( IS_DIRTY (infop) ) ? 'm' : '.';
238
239         sprintf(target, "%6ld %2d/%02d/%02d %2d:%02d:%02d - ",
240                 infop->fi_fsize,
241                 C_MONTH(infop), C_DAY(infop), C_YEAR(infop)-1900,
242                 C_HR(infop), C_MIN(infop), C_SEC(infop) );
243
244         while( *target )
245             target++;
246     }
247
248     if( IS_SUBDIR(infop) && graphics )
249         target = cpy( target, BOLDFACE );
250
251     target = cpy( target, pathname );
252     target = cptolower( target, infop->fi_name );
253
254     if( IS_SUBDIR(infop) && graphics )
255         target = cpy( target, ALL_OFF );
256
257     return( target - startstr );

```

(Continued on next page)



Number One In Performance

Hard Disk Intelligent VCR Backup for AT/XT/PC

FEATURES

- High speed microprocessor controlled backup (68000)
- Two channel interface
- Built in LAN channel
- Software control of most VCR functions including Fast Forward, Rewind, and auto backup using VCR timer capabilities
- Economical VHS or Beta formats



West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345
 East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450
 Distributor: Telemarketing Services, Inc.
 1897 Garden Ave., Eugene, OR 97403, 800-874-2288

Circle no. 174 on reader service card.

IEEE-488



GPIB \leftrightarrow PC

Hardware Flexibility

- Low cost for instrument control
 - 300K bytes per second
 - \$395 complete with software
- High performance data links
 - Maximum speed of GPIB
 - On-board buffering

Software

- Software specially tested for the IBM PC and compatibles
- Over \$1,000,000 in software development
- Easy to use, yet handles any GPIB application
- Works with Lotus 1-2-3
- UNIX, DOS and over 12 languages

Applications Support

- Applications Library with sample programs & TIPS for all major instruments
- Full staff of Applications Engineers dedicated to support your specific needs

Other IEEE-488 Products

- Interfaces & Software for
 - Multibus VMEbus
 - DEC Q-bus & UNIBUS
 - STD & S-100 bus
- General GPIB Products
 - GPIB Bus Testers
 - GPIB Bus Extenders
 - Stand-Alone Controllers

NATIONAL INSTRUMENTS

12109 Technology Boulevard
Austin, TX 78727
1 (800) 531-GPIB
In Texas (800) IEEE-488
Telex: 756737 NAT INST AUS

C CHEST

LISTING EIGHT (Listing continued, text begins on page 14)

```
258 }
259
260 /*-----*/
261
262 static int      add_entry( infop, dp, path )
263 FILE_INFO          *infop ;
264 register DIRECTORY    *dp ;
265 char                *path ;
266 {
267     /*      Add an entry to the DIRECTORY structure. Return 0 if
268     *      it was added, one if it wasn't.
269     */
270
271     char          buf[128] ;
272     register int    len ;
273
274     /*
275     *      If we're not printing hidden directories but the current
276     *      directory is nonetheless hidden, return immediately.
277     *      Similarly, return if the directory is full.
278     */
279
280     if( !dp->hidden && (IS_HIDDEN(infop) || *infop->fi_name == '.') )   return 1;
281
282     if( dp->maxdirs <= 0 )           /* No more room in dirv. return */
283         return 0;                   /* error status */
284
285     /*
286     *      Update the directory count or the file count as appropriate
287     *      return immedialy if we're looking at a file and we aren't
288     *      supposed to list file. The same with directories.
289     */
290
291     if( IS_SUBDIR(infop) )
292     {
293         if( dp->dirs )
294             dp->ndirs++ ;
295         else
296             return 1;
297     }
298     else
299     {
300         if( dp->files )
301             dp->nfiles++ ;
302         else
303             return 1;
304     }
305
306     /*
307     *      Convert the FILE_INFO structure to an ascii string and put
308     *      it into buf. Then malloc a chunk of memory the correct size,
309     *      copy the ascii string there, and put the malloced memory
310     *      into dirv at the correct place.
311
312     */
313
314     Longfmt = dp->longf;
315
316     len = dirtoa( buf, infop, dp->graphics, path );
317
318     if( len > dp->width )
319         dp->width = len ;
320
321     if( *dp->lastdir = malloc(len + 1) )
322     {
323         strcpy( *dp->lastdir++, buf ) ;
324
325         /* Add file size to total. Note that the actual amount
326         * of space (# of clusters) used by the file on the
327         * disk is used.
328         */
329
330         dp->nbytes += ROUND( infop->fi_fsize, Cluster_size );
331
332         --dp->maxdirs;
333         return 1;
334     }
335
336     fprintf(stderr,"Can't get memory for directory\n");
337     return 0;
338 }
339
340 /*-----*/
341
342
343 static void      copy_path( dest, src )
344 char          *dest, *src;
345 {
```

(Continued on page 62)

Circle no. 271 on reader service card.

MS-DOS, UNIX, APPLE MAC, CP/M, NETWORKS and MORE. ONE c-tree ISAM DOES THEM ALL!

The creator of Access Manager™ brings you the most powerful C source code, B+ Tree file handler: **c-tree™**

- multi-key ISAM and low-level B+ Tree routines
- complete C source code written to K&R standards
- single-user, network and multi-tasking capabilities
- fixed and variable record length data files
- virtually opened files accommodate limited file descriptors
- no royalties on application programs

\$395 COMPLETE

Specify diskette format:

- 5 1/4" MS-DOS
- 8" CP/M
- 3 1/2" Mac
- 8" RT-II



For VISA, MC and COD orders
call (314) 445-6833

FairCom
2606 Johnson Drive
Columbia, MO 65203

© 1985 FairCom

The following are trademarks: c-tree and the circular disk logo—FairCom; MS—Microsoft Inc.; CP/M and Access Manager—Digital Research Inc.; Unix—AT&T; Apple—Apple Computer Co.

Circle no. 93 on reader service card.

**THE BEST Z80
ASSEMBLER ON
THE MARKET JUST
GOT BETTER!**

Z80ASM

NOW ONLY \$49.95

**DON'T ASK HOW OURS CAN BE SO FAST . . .
ASK WHY THEIRS ARE SO SLOW!**

"... a breath of fresh air..."

Computer Language, Feb. 85

"... in two words, I'd say speed & flexibility",

Edward Joyce, User's Guide #15

Now fully compatible with M80 in .Z80 mode with many extensions. Time & date in listing, 16 char. externals, plus many other features.

To order, or to find out more about our complete family of development tools, call or write:

SLR Systems

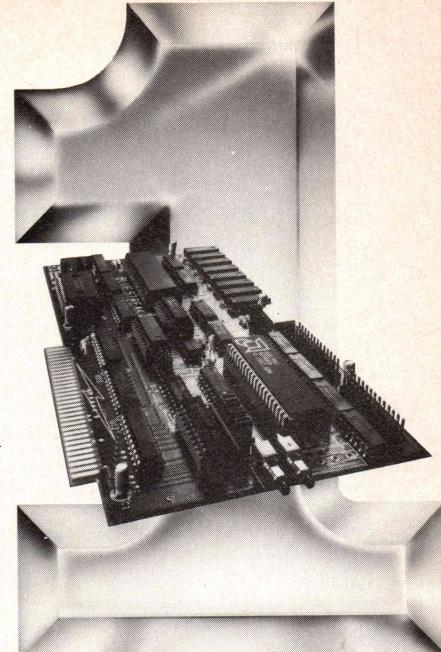
1622 N. Main St., Butler, PA 16001
(800) 833-3061, (412) 282-0864
Telex 559215 SLR SYS



C.O.D., Check or
Money Order Accepted

SHIPPING: USA/CANADA + \$3 • OTHER AREAS + \$10
Z80 CP/M compatibility required.

Circle no. 78 on reader service card.



**Number One
in Performance**

**Z80H
BLUE STREAK™**

**IBM/AT/XT/PC- 8mz
No Wait States**

FEATURES

- 64K-256K RAM
- 2K-8K EPROM/Static Ram
- 2 Serial Ports
Async/Sync/Bisync Communications
- Real Time Clock
- Memory-mapped Dual-port BUS
- On-board/Remote Reset NMI capability
- Up To 32 Boards Per AT/XT/PC
- Can Operate As Standalone Processor
- Less Than Full Size Board
(will fit other compatibles.)

SOFTWARE

- ZP/M tm CP/M Emulation Software
(Supports Most CP/M Software)
- Multiuser Capability If Used As A Slave Processor

IBM is a registered trademark of International Business Machines.
CP/M is a registered trademark of Digital Research Corp.

TLM Systems

West: 4704 W. Jennifer, Suite 105, Fresno, CA 93711, 209/276-2345
East: 67 Grandview, Pleasantville, NY 10570, 914/747-1450
Distributor: Telemarketing Services, Inc.
1897 Garden Ave., Eugene, OR 97403, 800-874-2288

Circle no. 175 on reader service card.

LISTING EIGHT (Listing continued, text begins on page 14)

```

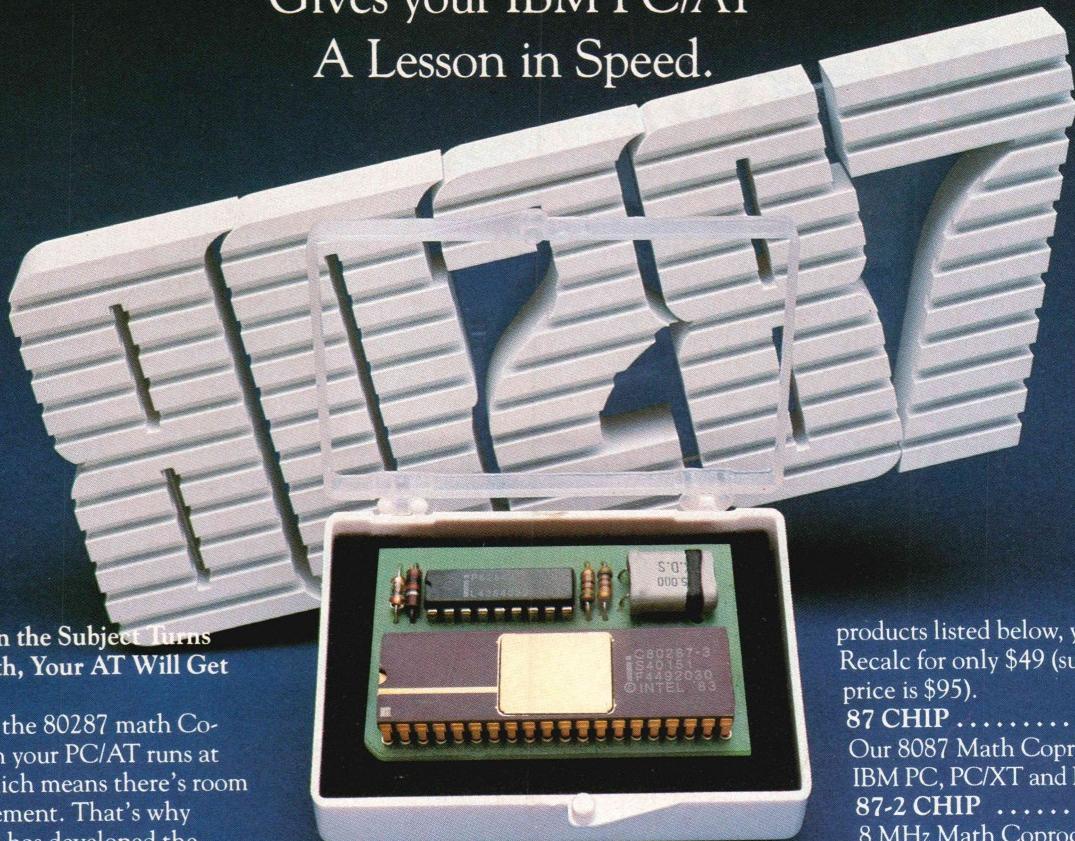
346     /*      Copy only the pathname part of the file spec contained in
347     *      src to dest. Path names longer than 64 characters are
348     *      truncated so dest must be at least 64 characters long.
349     */
350
351     register char  *p, *slash;
352
353     for( p = slash = src ; *p ; p++ )
354         if( *p == '/' || *p == '\\' || *p == ':' )
355             slash = p + 1;
356
357     for(p = src; p < slash  &&  p - src < 64 ; *dest++ = *p++ )
358         ;
359
360     *dest = 0;
361 }
362
363 /*-----*/
364
365 static void    clab( dest, src )
366 register char  *dest, *src;
367 {
368     for(; *src ; src++, dest++)
369         if( *src != '.' )
370             *dest = *src ;
371 }
372
373 /*-----*/
374
375 static int    cmp( pp1, pp2 )
376 char  **pp1, **pp2;
377 {
378     /*      Comparison routine needed for ssort()   */
379
380     register char  *p1 = *pp1;
381     register char  *p2 = *pp2;
382
383     if( Longfmt )
384     {
385         /*      Skip forward to the '-' that will precede
386         *      the filename.
387         */
388
389         while( *p1 && *p1 != '-' )
390             p1++;
391
392         while( *p2 && *p2 != '-' )
393             p2++;
394     }
395
396     return( strcmp(p1, p2) );
397 }
398
399 /*-----*/
400
401 DIRECTORY  *mk_dir( size )
402 register unsigned size;
403 {
404     /*      Make a DIRECTORY with the indicated number of dirv entries.
405     *      Note that since one dirv entry is declared as part of the
406     *      DIRECTORY header, we'll actually have size+1 entries
407     *      available, though the last one is never used. We allocate
408     *      it so that we can terminate the list with a null
409     *      entry, even if the list is full.
410     */
411
412     register DIRECTORY  *dp;
413
414
415
416
417     if( !( dp = (DIRECTORY *)calloc( (unsigned)1,
418                                     sizeof(DIRECTORY) + (size * sizeof(char *)) ) )
419         return 0;
420
421     dp->maxdirs = size ;
422     dp->lastdir = (char **) dp->dirv;
423     return dp;
424 }
425
426 /*-----*/
427
428 del_dir( dp )
429 register DIRECTORY  *dp;
430 {
431     /*      Delete a directory made with a previous mk_dir call.
432     *      Note that all the strings pointed to by dirv entries
433     *      are assumed to have been gotten from malloc (this is
434     *      always true if dir() is used to fill the strings.

```

(Continued on page 66)

HAUPPAUGE

Our 287 FAST/5
Gives your IBM PC/AT
A Lesson in Speed.



Now, When the Subject Turns to Fast Math, Your AT Will Get an A+.

Right now, the 80287 math Co-processor in your PC/AT runs at 4MHz. Which means there's room for improvement. That's why Hauppauge has developed the unique 287 FAST/5—a module that gives your AT an unmatched 5MHz capability. Speeding up your math computations by 25%. Now that's an A+ performance! To insure easy installation, we've created a carrier mount for our 287 FAST/5. Just plug the module into your PC/AT's 80287 Numeric Coprocessor socket. No expansion slot required.

1-2-3 GETS A LESSON IN SPEED, TOO!

Introducing Hauppauge's New Lotus Support

In addition to our 287 FAST/5, Hauppauge has recently developed a Lotus 1-2-3 support package that can accelerate your calculations up to 30 times. The package? Our 8087 coprocessor chip and Recalc™ software. Recalc links your Lotus 1-2-3 program to the fast math capability of our 8087 chip. By sending numeric operations to the 8087, computation time can be reduced from 60 seconds to as little as 2 seconds. And the more complex the calculation, the more time you save!

Affordability

We've raised the speed limit for Lotus 1-2-3 and lowered the cost of Recalc software. When you buy any of the Hauppauge

products listed below, you can get Recalc for only \$49 (suggested retail price is \$95).

87 CHIP \$129

Our 8087 Math Coprocessor for the IBM PC, PC/XT and PC compatibles.

87-2 CHIP \$195
8 MHz Math Coprocessor for Compaq Desk-Pro and Olivetti PC's.

287 CHIP \$219

80287 Math Coprocessor chip that runs at 4MHz in the IBM PC/AT.

287 FAST/5 \$249

Our 80287 Math Coprocessor enhancement module that boosts speed in the IBM PC/AT to 5MHz.

Now Available! The 287 FAST/8 \$379

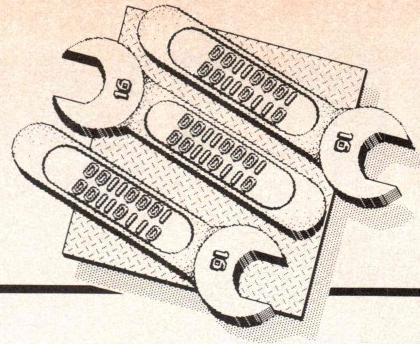
In addition to our 1-2-3 support, we're introducing our new 287 FAST/8—a module that gives your IBM PC/AT an unprecedented 8 MHz capability. Doubling the speed of your math calculations! (module pictured above)

Hauppauge's 287 FAST/5 and Lotus support package are teaching the IBM PC, PC/XT and PC/AT a thing or two about fast math. Learn more by calling us today—or contact your local computer dealer.

Hauppauge Computer Works

358 Veterans Memorial Highway, Suite MSI
Commack, New York USA 11725 (516) 360-3827
Available from your local computer dealer

**Z80 CP/M USERS TAKE HEART!
HERE'S ALL YOU NEED TO WRITE
YOUR OWN PROGRAMS FOR ONLY:
\$25.00!**



DR. DOBB'S Z80 TOOLBOOK

By David E. Cortesi

Do you use CP/M? Do you feel as if the only part of the computer industry that has not abandoned you is your own Z80 computer? It keeps on working, but when you need programs for it, you have to write

them yourself. When you do, you quickly find that while Pascal or Basic is okay for some things, there is often no substitute for the speed, small size, and flexibility of an assembly language program.

DR. DOBB'S Z80 TOOLBOOK puts the power of assembly language in the hands of anyone who's done a little programming. You'll find:

- **A method of designing programs and coding them in assembly language**—and—a demonstration of the method in the construction of several complete, useful programs.
- **A complete, integrated toolkit of subroutines** for arithmetic, for string-handling, and for total control of the CP/M file system. They bring the ease and power of a compiler's runtime library to your assembly language work, without a compiler's size and sluggish code.

Best of all, every line of the toolkit's source code is there for you to read, and every module's operation

is explained with the clarity and good humor for which Dave Cortesi's writing is known.

Order the Z80 Software on Disk! Save Yourself the Frustration of File Entry

All the software in **DR. DOBB'S Z80 TOOLBOOK**—the programs plus the entire toolkit, both as source code and as object modules for both CP/M 2.2 and CP/M Plus—is yours on disk. (A Z80 microprocessor and a Digital RMAC assembler or equivalent are required.)

Receive **DR. DOBB'S Z80 TOOLBOOK**, along with the software on disk, together for only \$40.

To order, return this form with your payment, plus \$1.75 for shipping in the U.S., \$3.75 outside the U.S., to: M&T Publishing, 501 Galveston Dr., Redwood City, CA 94063

Or, for faster service on credit card orders, CALL TOLL FREE 1-800-528-6050, ext. 4001. Refer to item #022 for the **Z80 Toolkit** and item #022A for the **Z80 Toolkit** with the disks. Please specify one of the disk formats offered below.

Send me _____ copies of **DR. DOBB'S Z80 TOOLBOOK**:

Send me _____ sets of **DR. DOBB'S Z80 TOOLBOOK** along with the software on disk for \$40 per set:

\$40 ea _____

Subtotal _____

(CA residents add applicable _____ % sales tax.)

Shipping _____

TOTAL _____

For the Z-80 software on disk, please specify one of the following formats:

_____ 8" SS/SD

_____ Osborne

_____ Apple

_____ Kaypro

Check enclosed _____ Charge my _____ Amer. Exp.

_____ VISA _____ M/C

CARD # _____ EXP. DATE _____

SIGNATURE _____

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

3113C



Announcing a radical new idea in PC-AT programming: **FREEDOM**

Alsys Ada Compiler for the 80286 Defeats the Tyranny of 640K DOS;
Liberates the Full 16MB Memory Capability of the Processor

The 80286 is a powerful chip. It can directly address up to 16 megabytes of memory. But unfortunately, you can't. DOS won't let you. And the compilers for whatever language you are currently using won't let you.

Until now.

Alsys has developed a new Ada compiler for the IBM PC-AT. Ada, of course, is the language mandated by the DoD for critical applications. Many believe it will be the dominant language for the rest of the eighties and nineties.

But leave aside Ada's virtues as a highly maintainable, portable, readable, software engineered language. Leave aside its acceptance and sponsorship by DoD, NASA, NATO, the FAA and large numbers of commercial users. Forget (if you can) the \$12 billion forecast in just DoD Ada sales through 1989.

Think only of a million plus lines of code running on a PC-AT! And think of the code executing *faster* than C or Pascal!

Think of the programs you could write if you could address 16 megabytes!!

It's like moving your AT from primitive to professional, roller skates to Rolls Royce. It lets you and your AT do everything you were meant to do.

The new Alsys Ada compiler, 300,000 lines of Ada code and self-compiled (with only 3 megabytes of memory!), also provides complete memory protection. An incorrect program affects no areas of memory except those allocated to the program. In particular, the operating system cannot be destroyed. And it does this, under control of DOS, *without any changes to DOS of any kind!*

No more Alt-Ctrl-Del restarts after a bug damages DOS!

Alsys is the premier Ada company in the world. France, U.S., U.K. And is about to become the premier AT compiler company in the world, too. For any language. For serious programmers frustrated by DOS.

Use the coupon now. Or Call. Freedom is a precious thing.



Alsys, Inc. • 1432 Main Street
Waltham, MA 02154 • U.S.A.
Phone: (617) 890-0030 • Telex: 948536

Alsys, Ltd. • Partridge Hse, Newton Road
Henley-on-Thames • Oxon RG91 EN, England
Phone: (0491) 579090 • Telex: 846508

Alsys, S.A. • 29, Avenue de Versailles
78170 La Celle St. Cloud • France
Phone: (3)918.12.44 • Telex: 697569

ALSYS, INC.,
1432 Main Street,
Waltham, MA 02154

Tell me more about a million lines

of code on an AT. Send me literature.

Call me. Tell me about prices, delivery,
warranties, support.

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

Phone () _____

SOFTWARE DESIGN ENGINEERS

We're looking for nothing less than the top Software Design Engineers in the country — wizards who can combine microcomputer software development skills with creative imagination. You will work in networking, sophisticated graphics, operating systems, compilers, productivity software, product marketing and other exciting areas. In each project, you'll have the opportunity to explore totally new realms of microcomputer software.

The right candidates will have a degree in Computer Science, or a related field with a minimum of 3 years relevant work experience in the microcomputer industry. Experience in micros, systems programming using "C", Pascal, or Assembler, operating systems and working in small teams is essential to these positions. Experience in 8086 Assembler, XENIX/UNIX, MS-DOS, and microprocessors (286, 8086, 86000) is desired.

Microsoft is located in the Seattle area of the beautiful Pacific Northwest, where the amenities of civilization live side by side with the grandeur of mountains and pine forests. It's one of the most prized living environments in the United States.

Come to Microsoft, where you will have the space and flexibility to prove to the computer industry that you're the best at what you do. Microsoft offers excellent opportunities and a complete benefits package. Send your resume to: **MICROSOFT CORPORATION, Human Resources, Dept. SS, 3055 112th NE, Box 97200, Bellevue, WA 98009.** An Equal Opportunity Employer.

MICROSOFT
The High Performance Software

DATALIGHT C

LARGE MODEL

The Datalight C compiler is a full-featured UNIX System 5 compiler for PC and compatible computers. Datalight C features fast compile time, DLC one-step compile command, a MAKE program, full 8087 and software floating point, and Lattice C compatibility. The library contains UNIX standard functions and PC specific functions with easy access to DOS / BIOS functions. The package includes source code for UNIX-like tools: cat, fgrep, diff wc, and pr.

\$60

The Developer's kit contains all features of the Datalight C compiler plus support for LARGE MEMORY MODEL, ROMable code, third-party debuggers and the complete source for all library functions and start-up code.

\$99

Datalight C provides tight, fast, production-quality code as shown in the following table (excerpt from "The State of C," PC-TECH Journal, January 1986, used with permission).

	Datalight	ECO	Lattice	Microsoft	Mark Wms.
Fibonacci	21.8	22.7	24.9	25.6	27.5
Sieve	21.0	24.8	21.0	26.3	23.3
Getc / Putc	24.0	73.9	62.2	51.3	62.2

Datalight

11557 8th Ave. N.E.
Seattle, Washington 98125
(206) 367-1803

VISA and MasterCard accepted. Add \$3 shipping (\$5 UPS BLUE). Outside USA add \$15. Washington State residents add 7.9% sales tax.

Requires MS-DOS 2.0 or later, 2 DSDD disks.

Lattice C, a trademark of Lattice Corp. MS-DOS, a trademark of Microsoft. UNIX is a trademark of Bell Labs.

Circle no. 203 on reader service card.

nine track tape users Micro to Mainframe Connection

The Model TC-50 ½-inch tape subsystem provides a standard medium for transmission of mainframe data base information to PC users, while maintaining mainframe isolation and data integrity. Use ODI subsystems to import data to data base programs like dBase III.

The TC-50 subsystem also provides fast back-up capability as well as a device driver and interface software for popular compilers.

The TC-50 subsystem includes tape drive controller, cables and documentation. All ODI products carry a 30 day unconditional money-back guarantee, and are warranted for one year, parts and labor.

ODI

Overland Data, Inc.

5644 Kearny Mesa Road
San Diego, CA 92111
Tel. (619) 571-5555
Telex 754923 OVERLAND

Also Available —
XENIX tape
subsystems
for the
IBM AT



Circle no. 192 on reader service card.

LISTING EIGHT (Listing continued, text begins on page 14)

```

524     firstdir = dp->lastdir;
525
526     /*      Now go look for the file:
527     */
528
529     if( !find_first(spec, ATTRIBUTES, &regs) )
530         if( !add_entry(&info, dp, path) )
531             goto abort;
532
533     if( haswild(spec) )
534         while( !find_next( &regs ) )
535             if( !add_entry(&info, dp, path) )
536                 goto abort;
537
538     if( dp->sort )
539         ssort( (char *)firstdir, dp->lastdir - firstdir,
540                sizeof(char*), cmp);
541
542 abort:
543     regs.x.ds = seg;           /* Restore the original disk */
544     regs.x.dx = off;          /* transfer address. */
545     DOSCALL( SETDATA, regs );
546 }

```

End Listing Eight

LISTING NINE

```

1 #include <stdio.h>
2 #include <ctype.h>
3
4 #define MAXARGC      (unsigned)128
5 #define isquote(c)    ((c)=='"' || (c)=='\'')
6
7 extern char    *getenv( char* );
8 extern char    *malloc( unsigned );
9
10 /*-----*/
11
12 static char    *nextarg( pp )
13 char        **pp;
14 {
15     register char   *p;
16     char        *start;
17     register int   term;
18
19     if( !(p = *pp) )
20         return (char *) 0;
21
22     while( isspace(*p) )
23         p++;
24
25     if( isquote(*p) )      /* Can't use a conditional because */
26         term = *p++;      /* of order of evaluation problems */
27     else
28         term = ' ';
29
30     for( start = p; *p ; p++)
31     {
32         if( *p == term && *(p-1) != '\\\'')
33         {
34             *p++ = '\0';
35             break;
36         }
37     }
38
39     *pp = p;
40
41     return start;
42 }
43
44 /*-----*/
45
46 int    reargv( argc, argvp )
47 char   ***argvp;
48 int    *argc;
49 {
50     register int   argc = 0 ;
51     register int   maxc = MAXARGC ;
52     char        **argv, **start_argv ;
53     char        *env, *p ;
54
55     if( !(env = getenv("CMDLINE")) || !*env )
56         return 0;
57
58     if( !(p = malloc( strlen(env)+1 )))
59         return 0;

```

(Continued on page 70)

New for 'C'

A "C" programmer's tool to increase screen development productivity for the IBM PC. Security checking and help screen display are available at both the screen and field level. The automatic conversion of data types, to and from ASCII screen format, and the many other productivity-oriented features, set ZVIEW apart from the rest.

Screen Painter Highlights:

- Border colors and all character attributes and colors are supported.
- Draw single or double lined boxes using preset key strokes.
- Two field sensitivity settings to facilitate the moving and adding of fields, without destroying existing field characteristics.
- Three types of fields are available: "Protected," "Unprotected" and "Heading." The number of fields is limited to 600!!
- Both 40 and 80 column screens are supported.

Optional Field Characteristics:

- Choose left or right justification, with zero or blank fill.
- Automatic key stroke conversion to upper or lower case.
- Edit fields to be numeric (signed or unsigned), decimal (zero to six decimals supported), alpha or alphanumeric.
- Display numerical values with or without commas inserted.
- All "C" data types are supported, including a special long value which is displayed as a decimal field.
- From and to range checking and character matching edit.
- Security level settings to restrict inquiring or updating of a field.
- Override ZVIEW's default tabbing sequence.
- Assignment of a single or multiple screen help file, to be displayed when the field level help key is pressed.
- Compare one field to three other fields on the current screen.

Program Interface Highlights:

- Only seven run-time library functions control all aspects of the program to screen interface.
- Dynamically change any field characteristic at run-time.
- A wide range of run-time variables to further customize ZVIEW's operation.
- One call to ZVIEW's "Waitkey" function, performs all field edits and program to end user interface.
- Automatic data conversion of all data types to and from data structures and buffers. Data goes directly from data type to screen format and back, with one call each way.

Requirements:

- Microsoft 3.0 "C" or Lattice "C" compiler. (Call for additional information on compiler availability).
- IBM PC, XT, AT or compatible, MS/PC DOS, one 320k drive, a color graphics adapter and any 40 or 80 column display.

Price:

- \$245
Includes manual and a detailed program example.



TO ORDER CALL TOLL FREE 1-800-423-0930

Customer Service and Nevada residents:
call 1-702-798-5910

IBM PC, XT, AT and PC-DOS are trademarks of International Business Machines.
MICROSOFT and MS-DOS are trademarks of Microsoft.
ZVIEW is trademark of Data Management Consultants



Or Write: Data Management Consultants
5325 So. Valley View Blvd. Suite #7
Las Vegas, NV 89118

Master Card, Visa or company
check accepted

AT LAST —

A handbook that contains the Programming Codes for 100's of popular printers.

Announcing:

PROGRAMMERS' HANDBOOK OF COMPUTER PRINTER COMMANDS

The handbook gives you:

- Codes for printers made by over 40 Printer Manufacturers.
- Easy to use spiral bound book of over 250 pages of Programming Codes written in table form.
- Codes arranged by Written Code, Hex and Decimal equivalent, and with a brief description of what each code does.
- Codes for either Daisy-Wheel or Dot Matrix Printers (models through 1984).

ONLY \$37.95 + \$2 shpg./hdlg. ppd.
with a two week approval guarantee. IF NOT SATISFIED, return in original carton for refund of book price only.

TO ORDER: 1-800-628-2828 ext. 534



Circle no. 246 on reader service card.

UNIVERSAL CROSS-REFERENCER

• IT WORKS WITH ALL LANGUAGES

BASIC, C, PASCAL, FORTRAN, COBOL, ASSEMBLER, dBASE . . . you name it.

• IT HANDLES standard languages, extended languages & exotic languages.

• IT CONFIGURES to your compiler, interpreter, or assembler — ALL BRANDS — ALL VERSIONS.

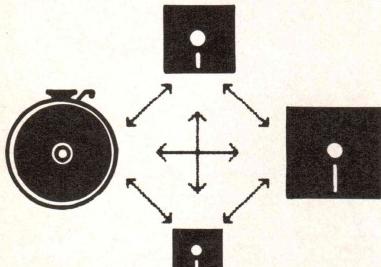
INTRODUCTORY **\$39.95**

\$3.00 S/H - MC/Visa/Check - 6% Texas Sales Tax
For the IBM PC, XT & compatibles. DOS 2.0+

DALSOFT SYSTEMS
3565 High Vista - Dallas, TX 75234
(214) 247-7695

Circle no. 86 on reader service card.

DATA CONVERSION



TRANSFER DATA BETWEEN OVER 500 DIFFERENT COMPUTER SYSTEMS

WORD PROCESSORS TOO

QUICK TURN-AROUND

PRICES FROM \$9 PER DISK

CALL OR WRITE FOR YOUR

FREE CATALOG

PORT-A-SOFT

555 S. STATE ST., SUITE #12
P.O. BOX 1685, OREM, UT 84057
(801) 226-6704

Circle no. 229 on reader service card.

C CHEST

LISTING NINE (Listing continued, text begins on page 14)

```

60     if( !(argv = (char **) malloc( MAXARGC * sizeof(char *) ) ) )
61         return 0;
62
63     strcpy(p, env);
64     start_argv = argv;
65     for( maxc=MAXARGC; --maxc >= 0 && (*argv++ = nextarg(&p)); argc++)
66     ;
67
68     if( maxc < 0 )
69         fprintf(stderr, "Command line truncated\n");
70
71     *argc = argc;
72     *argv = start_argv;
73     return 1;
74 }
75 }
76 */
77
78
79 #ifdef DEBUG
80
81 main( argc, argv )
82 char    **argv;
83 {
84     printf("Original command line is: |");
85     while( --argc >= 0 )
86         printf("%s|", *argv++ );
87
88     if( !reargv( &argc, &argv ) )
89         printf("\nCMDLINE not present\n");
90     else
91     {
92         printf("New argc = %d\n", argc );
93         printf("\nModified command line is: |");
94
95         while( --argc >= 0 )
96             printf("%s|", *argv++ );
97
98         printf("\n");
99     }
100 }
101#endif
102
```

End Listing Nine

LISTING TEN

```

1 /*           SSORT.C      Works just like qsort() except that a shell
2  *                                sort, rather than a quick sort, is used. This
3  *                                is more efficient than quicksort for small numbers of elements
4  *                                and it's not recursive so will use much less stack space.
5  *
6  *                                Copyright (C) 1985, Allen I. Holub. All rights reserved
7  */
8
9 void ssort( base, nel, width, cmp )
10 char *base;
11 int nel, width;
12 int (*cmp)();
13 {
14     register int i, j;
15     int gap, k, tmp;
16     char *p1, *p2;
17
18     for( gap = nel >>1 ; gap > 0 ; gap >>=1 )
19         for( i = gap; i < nel; i++ )
20             for( j = i-gap; j >= 0 ; j -= gap )
21             {
22                 p1 = base + ( j * width );
23                 p2 = base + ((j+gap) * width );
24
25                 if( (*cmp)( p1, p2 ) <= 0 )
26                     break;
27
28                 for( k = width; --k >= 0 ; )
29                 {
30                     tmp = *p1;
31                     *p1++ = *p2;
32                     *p2++ = tmp;
33                 }
34
35 }
```

End Listings

**Includes complete source
code and documentation!
Only \$29.95 Each!**

A Unix-like Shell AND a Unix-like Utility Package

by Allen Holub

Both Available on Disk for MS-DOS

THE SHELL

SH—A Unix-like Shell for MS-DOS

SH is an implementation of the most often used parts of the Unix C shell. This package includes an executable version of the shell along with the complete source code and full documentation.

Supported features are:

Editing

Command line editing with the cursors is supported. The line is visible as you edit it.

Aliases

Can be used to change the names of commands or as very fast memory resident batch files.

History

The ability to execute a previous command again. The command can be edited before being executed.

Shell variables

Macros that can be used on the command line.

Nested batch files

A batch file can call another batch file like a subroutine. Control is passed to the second file and then back to the first one when the second file is finished. DOS doesn't have this capability.

Unix-like syntax

Slash (/) used as a directory separator, minus (-) as a switch designator. A 2048 byte command line is supported. Command line wild card expansion. Multiple commands on a line.

The shell also supports redirection of standard input, standard output, and standard error.

This version corrects several bugs found in the original version printed in Dr. Dobb's Journal, December 1985 through March 1986 issues. It runs on any MS-DOS computer.

/util

A Unix-like Utility Package for MS-DOS

/util is a collection of Unix utility programs for MS-DOS. This package includes complete source code. All programs (and most of the utility subroutines) are fully documented. You'll find executable versions of:

cat

A file concatenation and viewing program

cp

A file copy utility

date

Prints the current time and date

du

Prints amount of space available and used on a disk

echo

Echoes its arguments to standard output

grep

Searches for a pattern defined by a regular expression.

ls

Gets a sorted directory.

mkdir

Creates a directory

mv

Renames a file or directory. Moves files to another directory.

p

Prints a file, one page at a time.

pause

Prints a message and waits for a response.

printenv

Prints all the environment variables.

rm

Deletes one or more files.

rmdir

Deletes one or more directories.

sub

Text substitution utility. Replaces all matches of a regular expression with another string.

To order, return this coupon with payment to: M&T Publishing Inc., 501 Galveston Dr. Redwood City, CA 94063

YES!

Please send me _____ copies of The Shell for \$29.95 each _____

copies of /util for \$29.95 each _____

SUBTOTAL _____

CA residents add applicable sales tax _____ % _____

Please add \$1.75 per disk for shipping _____

TOTAL _____

Check/Money Order enclosed

Please charge my _____ VISA _____ M/C _____

Amer. Exp.

Exp. Date _____

Card # _____

Name _____

Address _____

City _____

State _____ Zip _____

3113D

THE PROGRAMMER'S SHOP™

helps save time, money and cut frustrations. Compare, evaluate, and find products.

SERVICES

- Programmer's Referral List
- Dealer's Inquire
- Compare Products
- Newsletter
- Help find a Publisher
- Rush Order
- Evaluation Literature FREE
- Over 700 products
- BULLETIN BOARD - 7PM to 7AM 617-826-4086

Free Literature - Compare Products

Evaluate products. Compare competitors. Learn about new alternatives. One free call brings information on just about any programming need. Ask for any "Packet" or Addon Packet AI ADA. Modula BASIC C COBOL Editors PORTH FORTRAN PASCAL UNIX/PC or Debuggers, Linkers

AI - Expert System Dev't

Arity System - incorporate with C programs, rule & inheritance PC \$295
 1st Class - by example, interfaces \$250
 ExpertEASE - Develop by describing examples of how you decide. PC Call
 EXSYS - Thorough, improved. Debug, report gen. PC \$349
 INSIGHT1 - Probabilities, required thresholds, menus, fast (\$79). 2 adds backward, forward, partitions, dB2, lang, access. PC \$399
 Others: APES (\$359), Advisor (\$949), ES Construction (\$100), ESP (\$845), Experteach (\$399), Expert Choice (\$449)

AI-LISP

List Our

GC LISP - "Common", rich. Interpreter - Interactive Tutorial \$495 Call
 LARGE Model - 2 to 15 meg. Compiler and Interp. \$1190 Call
 TLC LISP - "LISP-Machine" - like, all RAM, classes, turtle graphics, 8087, compiler. CPM-86, MS \$225
 WALTZ LISP - "FRANZ LISP" - like, big nums, debug, CPM-80 MS \$149
 Others: IQ LISP (\$155), BYSO (\$125), MuLISP-86 (\$199)

AI-PROLOG

ARITY Standard - full, 4 Meg PC \$ 95
 Interpreter - debug, C, ASM PC \$ 350
 COMPILER/Interpreter - EXE PC \$ 795
 With Exp Sys, Screen - KIT PC \$1250
 MicroProlog - enhanced \$ 229
 MProlog - Improved. Faster PC \$ 725
 Professional MicroProlog MS \$ 359
 TransPROLOG - Learn Fast, Standard, tutorials, samples MS Call
 Others: Prolog-1 (\$359), Prolog-2 (\$1895).

AI-OTHER

METHODS - SMALLTALK has objects, windows, more PC \$239
 QNIAL - Combines APL with LISP Library of sample programs included. Source or binary. PC \$375
 SNOBOL4 + -great for strings, MS \$ 85

FEATURE

TransLisp - "Common subset, tutorial, editor, PP, trace. Best to learn. 20 Demos All MS Only \$ 75

BASIC

ACTIVE TRACE, DEBUGGER - BASICA, MBASIC, interactive, well liked MS \$ 79
 BASIC DEVELOPMENT SYSTEM - (BDS) for BASICA; Adds Renum. crossref, compress. PC \$115
 BetterBASIC all RAM, modules. structure. BASICA - like PC \$169
 8087 Math Support \$ 89
 Run-time module \$459
 CADSAM FILE SYSTEM - full ISAM in MBASIC source. MS \$ 75
 CB-86 - DRI CPM86, MS \$419
 Data Manager - full source MS \$325
 InfoREPORTER - multiple PC \$115
 PC/BASIC for Macintosh - by Pteradactyl. Compiles IBM BASICA, and MS BASIC for MAC syntax. \$250
 Prof. Basic - Interactive, debug PC \$ 89
 8087 Math Support \$ 47
 QuickBASIC by Microsoft - Compiles full syntax of IBM BASICA, 640K, PC \$ 85
 TRUE Basic - ANSI PC \$109
 Run-time Module PC \$459

COBOL

Macintosh COBOL - Full MAC \$459
 MBP - Lev II, native MS \$885
 MicroFocus Prof. - full PC Call
 Microsoft Version II - upgraded. Full Lev. II, native, screens. MS \$500
 Realia - very fast MS Call
 Ryan McFarland - portable MS \$695

Editors for Programming

BRIEF Programmer's Editor - undo, windows, reconfigure PC Call
 C Screen with source 80/86 \$ 75
 EMACS by UniPress - powerful, multiframe, windows, DOS, MLISP, programming. Source: \$949 \$299
 Entry Systems for C PC \$325
 Epsilon - like EMACS, full C-like lang. PC \$169
 FirsTime by Spruce - Improve productivity. Syntax directed for Turbo (\$69), Pascal (\$229), or C (\$239)
 Kedit - like XEDIT PC \$115
 PMATE - power, multitask 80/86 \$159
 VEDIT - well liked, macros, buffers, CPM-80-86. MS PC \$119
 XTC - multitasking PC \$ 95

Ask about ATARI ST, Amiga

RECENT DISCOVERY

Dan Bricklin's Demo Program - Prototype quickly, with realism. User feedback without programming. All 250 ASC characters plus attributes. Subsetting, building blocks, macros, thorough. PC \$ 75

C Language - Compilers

BDS C - solid value, fast CPM80 \$125
 C86 by CI - 8087, reliable MS Call
 Consulair Mac C w/toolkit MAC \$299
 ECO C/88 MS \$ 59
 Lattice C - from Lifeboat MS \$289
 Lattice C - from Lattice MS \$339
 Mark Williams - debugger MS \$379
 Megamax - tight, full ATARI/ST \$179
 Microsoft C 3.0 - new, MS \$259
 Q/C 88 by Code Works - Compiler source, decent code, cross MS \$125
 Wizard C - Lattice C compatible, full sys. III, lint, fast. MS \$379

C Language - Interpreters

C-terp by Gimpel - full K & R., OBJ and ASM, large progs. MS \$249
 INSTANT C - Source debug, Edit to Run-3 seconds MS \$399
 Interactive C by IMPACC Associates. Interpreter, editor, source debugger, profiler. PC \$395
 Introducing C - Interactive C to learn fast, tutorial PC \$115
 Professional Run/C has C plus ability to create add-in libraries, (Lattice C compatible) and load/unload them. MS \$199
 Run/C - improved MS \$109

C Libraries - General

Blaise C Tools 1 (\$109), C Tools 2 PC \$ 89
 C Food by Lattice - ask for source PC \$119
 C*LIB by Vance PC \$129
 C Utilities by Essential - Comprehensive screen graphics, strings, file handling, memory mgmt. Source. MS \$139
 Entelekon C Function Library PC \$119
 Entelekon C Windows PC \$119
 Entelekon Superfonts for C PC \$ 45
 Greenleaf Functions - portable, ASM \$139
 Polytron - for Lattice, ASM source \$ 99
 Software Horizons - Pack 1 \$129

C Libraries - Communications

Asynch by Blaise PC \$149
 Greenleaf - full, fast PC \$139
 Software Horizons - pack 3 PC \$119

C Libraries - Files

FILES: C Index by Trio - full B + Tree, vary length field, multi compiler
 /File is object only MS \$ 89
 /Pro is partial source MS \$179
 /Plus is full source MS \$349
 CBTREE - multiuser record locking, sequential, source, no royalties MS \$ 99

METRIC MINIMIZER

LISTING ONE (Text begins on page 24)

```

/* GLOBAL.H
 *
 * Global include file for the Variable Metric Minimizer program.
 *
 * (c) Copyright 1985, Billybob Software. All rights reserved.
 * This program may be reproduced for personal, non-profit use only.
 */

#define C80

/*
 * remove the above line if you are not using C/80
 */

#ifndef C80
/*
 * required for the Software Toolworks C/80 Version 3.0 compiler
 */

#define double float
#include "fprintf.h"
#include "scanf.h"

/*
 * the following is the <math.h> for C/80
 */

float sin(), cos(), atan(), sqrt(), exp(),
      pow(), pow10(), ln(), log(), fabs(), atof() ;
#endif

/*
 * other compilers
 */
#include <stdio.h>
#include <math.h>
#include <cctype.h>

```

```

/*
 * ctype.h may be needed for an isspace used in read.c
 * some systems need it, some don't!
 * check your math.h for fabs and atof declarations!
 */

#endif

double dot() ;
struct bound
{
    int fl ;
    double up ;
    double lo ;
    double mi ;
};

struct xydata
{
    double x ;
    double y ;
};

#define BOUND DATA
#define VECMAX 10
#define MATMAX 2
#define NEPS 0
#define STDES 1
#define DFP 2
#define BFGS -1
#define ALLDONE -3000
#define NEGEDM -2000
#define BADMIN -1000
#define BADLS 0
#define MAXITERS 1000
#define OK 2000
#define EDM1 3000
#define EDM2 TOOSML

struct bound
{
    struct xydata
}
```

End Listing One

(Continued on page 76)

WORLD'S FIRST ASSEMBLER INTERPRETER

Advanced Trace86™ 2.0

- Create .COM programs with BASIC-like commands: LOAD, SAVE, EDIT, LIST, RUN. Reloadable into AT86, complete with all labels, comments & variable names.
- Macro Assembler support-scan .MAP and .LST files for labels and variable names
- Full-screen symbolic tracing of any program.
- Powerful conditional breakpoint facilities, including hardware "button" support (if installed)
- "Screen save" option, or use dual monitors
- Hex/decimal calculator & converter
- Best 8087/80286/80287 support on the market!
- And more . . . Priced at \$175.00

To order or request more information contact:



Morgan Computing Co., Inc.
(214) 245-4763

P.O. Box 112730, Carrollton, TX 75011

Circle no. 128 on reader service card.

¿C? ¡SÍ!

If you're a C programmer (or want to be one), we speak your language. Subscribe to **The C Journal** today, and start increasing your productivity right away. We give you information you can **use** on **any** machine — IBM PC™, UNIX™-based, Macintosh™, or CP/M™ — micro, mini, or mainframe.

- in-depth reviews and feature articles — C compilers, editors, interpreters, function libraries, and books.
- hints and tips — help you work **better** and **faster**.
- interviews — with software entrepreneurs that **made it** — by using C.
- news and rumors — from the ANSI standards committee and the industry.

Limited Time Offer

Join our thousands of subscribers at the **Discount Rate** of only \$18 for a full year (regularly \$28)! Call us now at (201) 989-0570 for faster service — don't miss a single issue of **The C Journal**!

Please add \$9 for overseas airmail.

Trademarks — CP/M: Digital Research Inc. IBM PC: IBM Corp. Macintosh: Apple Computer Corp. The C Journal: InfoPro Systems. UNIX: AT&T Bell Labs.



InfoPro Systems
3108 Route 10
Denville, NJ 07834
(201) 989-0570



Circle no. 194 on reader service card.

C Programmers: Here are 8 ways You can be more productive

Dear Microcomputer Programmer,

Let me tell you how you can find and choose the best development software for your needs — software that will help you:

- * Speed your development efforts
- * Write even better programs

- * Increase productivity
- * Reduce your programming frustration

All you have to do is consider one of these eight products for C programmers (or the 97 other C compilers, interpreters, support libraries, debuggers, or addons we offer). Then call one of our knowledgeable consultants - toll free - for details, comparisons, or for our specially prepared packets on C, C Libraries, or C Productivity Tools.

There is no obligation. You risk nothing with our moneyback guarantee of satisfaction.

Yours for more productive programming,

— Bruce W. Lynch, President

First Aid for C Programs C ToolSet

Save time and frustration when analyzing and manipulating C programs.

DIFF and CMP-for "intelligent" file comparisons.

XREF - cross references variables by function and line.

C Flow chart - shows what functions call each other.

C Beautifier - make source more readable.

GREP - search for patterns.

PP - formats your code so that it is easier to read and understand.

C Util - acts as a general purpose file filter.

There are several other programs for converting and printing programs.

Portable. Full source code.
CPM, MSDOS \$135

Even for Small Files: Convenient, Fast Access

CBTREE — Only \$99

Why spend time writing file management code when you can use consistent, flexible, documented, professional function? Even multiuser record locking and variable-length records are supported.

Full, balanced Btree support includes use of multiple keys, unlimited number and length of keys.

Use this powerful ISAM, even if you've previously done without.

Learn how to write systems for managing large files by using CBTREE source as a guide. Modify it and transfer it to another operating environment without royalties.

SORT/MERGE Files for Clean, Fast Maintenance

with OPT-TECH SORT

Performance should not suffer with DOS or other "free" sorts. ISAMs alone are slow when 10% or even less is changed/added.

OPT-TECH includes:

- CALLable and Standalone use
- C, ASM, BAS, PAS, FTM, COBOL
- Variable and fixed length
- 1 to 9 fields to sort/merge
- Autoselect of RAM or disk
- Options: dBASE, BTrieve files
- 1 to 10 files input
- No software max for # Records
- All common field types
- By pass headers, limit sort
- Inplace sort option
- Output = Record or keys

Try what you're using on an XT: 1,000 128 byte records, 10 byte key in 33 seconds. MSDOS \$85

Add Communications Features to Your Programs

Greenleaf Comm Library

Greenleaf now enables you to communicate with remote systems or databases with an asynchronous communications library for C.

Individual transmission and reception ring buffers combine with an interrupt driven system. This eliminates the extra function of separately calling up the communications program.

Included are 1 library/object files, 100 functions; 100 page manual, complete source code, library tailor-made to suit compiler and memory. Hayes-compatible modem commands, and a complete sample file transfer program.

MSDOS \$149

Get File Access with TIGHTER Control

db_VISTA Data Management

Full source, no royalties and "normal" indexed file management are part of db_VISTA. Get more for the price of only an ISAM.

You can minimize data stored and access records even faster and more logically than just using indexes. Example: address and transaction data should not require redundant storage of customer names or numbers. Use pointers. Related data fields point to other related groups - the "network model" of data.

Use db_VISTA as a "normal ISAM" or save programming time, access time and file size. Lattice, C86, Williams, Desmet, Microsoft C.

MSDOS Multiuser source \$995, Object \$495

Single user source \$450, Object \$169

Unix, Xenix, & Macintosh versions also available. Call for details.

Make REAL TIME Programming Practical

Csharp Realtime Toolkit

Data acquisition, process control, robotics and devices monitoring applications become practical with Csharp!

Full source code helps tailor programs to various boards and applications.

Reentrant, interrupt handling routines help schedule and react. Fast graphics routine's help visualize what is happening.

Control multiple ports reliably, schedule tasks based on events, manage priorities—all with modular, tested, and reliable routines.

Assess and manage the state of hardware at the object level. Let Csharp handle the details.

Portable C source supports RT11 UNIX and MSDOS \$600

Shorten Development Time, Cut Frustrations

BRIEF, The Programmer's Editor

Compile within BRIEF; use autoindent; "templates", C specific error support . . . use windows, UNDO, and multiple large files.

But edit YOUR WAY.

Every feature you'd expect and more are integrated elegantly - and it can be modified by you.

You deserve:

"...the best text editor you can buy." - John Dvorak, InfoWorld, 7/8/85

"...the best code editor . . ." David Irwin, Data Based Advisor, 8/85

PCDOS \$Call

Fast File Access with Source Variable Length Fields Save Space

Index ISAM Product Line

C-Index contains a high performance ISAM, balanced B + Tree indexing system and variable length fields. The result is a complete data storage system to eliminate tedious programming and add efficient performance to your programs.

Features include random and sequential data access, virtual memory buffering, and multiple key indexes.

With no royalties for programs you distribute, full source code, and variable length fields C-Index/Plus fits what you are likely to need.

Save time and enhance your programs with C-Index/Plus. MSDOS \$349. With C-Index/File for \$89, or/Pro for \$179.

If you call for our advice, you must be completely satisfied with the product you purchase from The Programmer's Shop. If not, you will receive a refund or replacement. Call now for details or our new catalog.

Circle no. 141 on reader service card.

THE PROGRAMMER'S SHOP

128 Rockland Street

Hanover, Massachusetts 02339

800-421-8006

In Mass. 800-442-8070 617-826-7531

METRIC MINIMIZER

LISTING TWO (Listing continued, text begins on page 24)

```
/* VMM.C
 *
 * The main driver routine, function library routine
 * and all library functions for vmm.
 * (c) Copyright 1985, Billybob Software. All rights reserved.
 * This program may be reproduced for personal, non-profit use only.
 */
#include "global.h"

#define DMAX 50      /* Size of experimental data array */
#define NFUN 3       /* Num of functions in function library */

/********************************************/

main()
{
    static int nparam;           /* number of parameters */
    static double const[VECMAX]; /* constants used in function */
    static double param[VECMAX]; /* parameters to be found */
    static double dstep[VECMAX]; /* step sizes for derivatives */
    static double epsilon[VECMAX]; /* stopping criteria */
    static BOUND limit[VECMAX]; /* limits if constrained */
    static char *pname[VECMAX]; /* parameter names */
    static double g0;             /* expected value of minimum */
    static int debug;            /* debug level, 0 if off */
    static int itermin;          /* minimum number of iterations */
    static int iterlim;          /* maximum number of iterations */
    static int nreset;           /* reset y after this many iters */
    static DATA exp[DMAX];       /* xy experimental data */
    static int npoints;          /* number of data points */
    static double (*fun)();        /* ptr to function to minimize */
    static int iters;             /* number of iterations it took */
    static int retcode;           /* return code from minimization */
    static double gmin;            /* the value at the minimum */
    static int method;             /* method used to update y */
    static int control;
    int i, j;

    for ( i=1 ; ; ++i )
    {
        raz ( &nparam, &itermin, &iterlim, &nreset, &debug,
              limit, dstep, param, &g0, epsilon, &method );

        if( (control = reader ( &fun, const, &nparam, param,
                               limit, dstep, pname, epsilon, &itermin,
                               &iterlim, &nreset, &g0, &debug, exp,
                               &npoints, DMAX, &method)) == ALLDONE )
            break;

        if ( control )
            continue;

        if ( dfault ( nparam, &itermin, &iterlim, &nreset,
                      epsilon, limit, dstep, debug ) )
            continue;

        retcode = minim ( nparam, fun, param, dstep, limit,
                          g0, itermin, iterlim, epsilon, &gmin,
                          &iters, debug, nreset, exp, npoints,
                          const, method );

        printf("\t*****\n");
        printf("\t* results from dataset %2d *\n", i );
        printf("\t*****\n");
        printf("stopped after %ld iterations, ", iters );

        switch ( retcode )
        {
        case NEGEDM : printf("negative e.d.m.\n"); break;
        case BADMIN : printf("new min > last min\n"); break;
        case BADLS : printf("line search failed\n"); break;
        case MAXITERS: printf("maximum iterations\n"); break;
        case EDM1   : printf("e.d.m. within tol.\n"); break;
        case EDM2   : printf("e.d.m. within tol.\n"); break;
        case TOOSML : printf("change too small\n"); break;
        default     : printf("unknown reason!!!\n"); break;
        }

        printf("value of the minimum was %11.4e\n", gmin );
        printf("parameter values at minimum:\n");
    }
}
```

(Continued on page 78)

Now You Know Why **BRIEF** is **BEST**

"BRIEF, The Programmer's Editor, is simply the best text editor you can buy." John Dvorak, INFOWORLD 7/8/85

The Program Editor with the **BEST** Features

Since its introduction, BRIEF has been sweeping programmers off their feet. Why? Because BRIEF offers the features **MOST ASKED FOR** by professional programmers. In fact, BRIEF has just about every feature you've ever seen or imagined, including the ability to configure windows, keyboard assignments, and commands to **YOUR** preference. One reviewer (David Irwin, DATA BASED ADVISOR) put it most aptly, "(BRIEF)...is quite simply the best code editor I have seen."

COMPILER SUPPORT

No matter what compiler you have, it will run inside BRIEF. If errors occur during compilation, the supplied macros place your cursor on the line with the first problem and display the compiler's message. After you make your corrections, you skip to the next error with one keystroke. BRIEF automatically moves your cursor to the right place, even if you've added or deleted lines.

BRIEF is preconfigured (using the built-in macro language) for the Microsoft Macro Assembler v 4.0, and the Microsoft, Computer Innovations, Lattice, and Wizard C compilers. If you use another product, you can modify the macros to support it.

Every Feature You Can Imagine

Compare these features with your editor (or any other for that matter).

- FAST
- Full UNDO (N Times)
- Edit Multiple Large Files
- Compiler-specific support, like auto indent, syntax check, compile within BRIEF, and template editing
- Exit to DOS inside BRIEF
- Uses all Available Memory
- Tutorial
- Repeat Keystroke Sequences
- 15 Minute Learning Time
- Windows (Tiled and Pop-up)
- Unlimited File Size -(even 2 Meg!)
- Reconfigurable Keyboard
- Context Sensitive Help
- Search for "regular expressions"
- Mnemonic Key Assignments
- Horizontal Scrolling
- Comprehensive Error Recovery
- A Complete Compiled Programmable and Readable Macro Language
- EGA and Large Display Support
- Adjustable line length up to 512

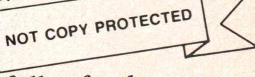
Program Editing YOUR Way

A typical program editor requires you to adjust your style of programming to its particular requirements - NOT SO WITH BRIEF. You can easily customize BRIEF to your way of doing things, making it a natural extension of your mind. For example, you can create ANY command and assign it to ANY key - even basic function keys such as cursor-control keys or the return key.

The Experts Agree

Reviewers at BYTE, INFOWORLD, DATA BASED ADVISOR, and DR. DOBB'S JOURNAL all came to the same conclusion - **BRIEF IS BEST!**

Further, of 20 top industry experts who were given BRIEF to test, 15 were so impressed they scrapped their existing editors!



MONEY-BACK GUARANTEE

Try BRIEF (\$195) for 30 days - If not satisfied get a full refund.
TO ORDER CALL (800-821-2492)

Solution Systems™

SOLUTION SYSTEMS, 335-D WASHINGTON ST., NORWELL, MA 02061, 617-659-1571

BRIEF is a trademark of UnderWare

METRIC MINIMIZER

LISTING TWO (Listing continued, text begins on page 24)

```
for (j=0 ; j<nparam ; ++j)
    printf("%15s %11.4e\n", pname[j] , param[j]);
printf("*****\n");
}

/*
 *      Function library. Note that NFUN is #defined at the top of this
 *      module.
 */

funlib ( funname , fun , np , nc , pname )
char   funname[] ;           /* name of function to be minimized      */
int    *fun ;                /* address of function (non-portable)   */
int    *np ;                 /* number of parameters in the function */
int    *nc ;                 /* number of constants in the function  */
char   *pname[] ;           /* parameter names vector               */
{
/*
 * use funname to return fun and np, nc from a list
 * for each new function in the library, you must:
 *      (1) declare the function "double" and link it in the load module
 *      (2) declare the function below under "functions available"
 *      (3) add an entry in the function table,including parameter names
 *      (4) add an entry in the switch table in the code
 *      (5) update the value of NFUN
 *
 * Available functions are:
 */
double  cohen();
double  sinus();
double  rosen();

/*
 * function table
 */
static struct
{
    char *name ;           /* function name in funname call      */
    int  nnpp ;            /* number of parameters in the function */
    int  nncc ;            /* number of constants in the function */
    char *ppnn[VECMAX]; /* list of names for the parameters   */
} fctlib[NFUN] = {
/* 0 */    { "cohen" , 2, 0 , "alpha" , "beta" },
/* 1 */    { "sine" , 4, 0 , "P0" , "P1" , "P2" , "P3" },
/* 2 */    { "rosen" , 10, 5 , "x1" , "x2" , "x3" , "x4" , "x5" ,
              "x6" , "x7" , "x8" , "x9" , "x10" }
};

int     j, k;

for ( j=0 ; j<NFUN ; ++j )
    if ( ! strcmp( funname , fctlib[j].name ) )
        break;

if (j==NFUN)
    return( 1 );

*np = fctlib[j].nnpp ;
*nc = fctlib[j].nncc ;

for (k=0 ; k < *np ; ++k)
    pname[k] = fctlib[j].ppnn[k] ;

switch( j )
{
case 0 : *fun = cohen ; break ;
case 1 : *fun = sinus ; break ;
case 2 : *fun = rosen ; break ;
}

return( 0 );
}

/*
 *      double cohen ( const , xx , data , npoints , user )
 *      double const[] ;          /* vector of constants used by this fcn */
 *      double xx[] ;            /* the current values of the vector x */
 */
```

(Continued on page 80)

MARCH Madness

NEW HOURS
7AM-6PM
P.S.T.

YOU LIKE IT. OR WE TAKE IT BACK!

If for any reason, you are not satisfied with any product you purchase, you may return it within 10 days of receipt for replacement, credit or a full refund.*

WE'LL PAY YOU IF YOU FIND A LOWER PRICE!

If you buy any item from us at pricing in this ad and find a lower price from any source in this issue, that has the identical product in stock, we'll not only refund the difference you paid, but also pay you 20% of the difference for your trouble! If you find a lower price in this issue before you buy, from any source that has the identical product in stock, we'll beat it!**

GUARANTEED AVAILABILITY!

Any item you order will be shipped within two working days or you will be given a firm ship date when you order! If for any reason we cannot ship by the date you are given, we will deduct 5% from the price of the products shipped late and credit your order accordingly.**

COMPONENTS

Quality Japanese mfg. from companies like OKI, HITACHI, TOSHIBA and FUJITSU.

256K DRAMS	Set of 9-150ns	\$26.46
64K DRAMS	Set of 9-150ns	\$8.91
8087-3	\$93.00	
8087-2	\$129.60	
80287	\$178.00	
27128	\$2.90	
V20-5MH replaces 8088	\$11.00	
27256	\$4.50	
2764	\$1.98	

MULTITECH MULTIFUNCTION BOARD

AST Sixpack™ compatible. Serial port, Parallel port, Game port, Clock/Calendar, RAMDISK and PSPOOL software. Up to 384 K memory expansion (empty)

\$99

1 Year Warranty

Inside
California

1-800-358-8881

1-800-826-3736

Outside
California

*THE FINE PRINT

Excluding software. Prior return authorization required; all items returned must be in original condition with carton, packing and all manuals, etc. We accept Cashier and Certified checks, Money Orders, personal checks (product shipped when check clears), VISA and MasterCard with no surcharge and American Express (4% surcharge). All products shipped UPS ground unless specified otherwise (freight FREE for orders over \$100). All normal manufacturer's warranties apply.

FREE DIGITAL WATCH

With the purchase
of any disk drive in
this issue, we'll
include a 7 melody
alarm, Quartz
chrono watch FREE!
(limit one per
customer)

COMPLETE INTERNAL HARD DRIVE SYSTEMS

Includes drive, PC/XT controller card, cables and install procedures. Capacities listed are unformatted. We sell only the finest drives from Seagate, Mitsubishi, Microscience, and others guaranteed to meet or exceed original manufacturer's specifications.

13MB	1/2 Ht. 85ms	\$388.
25MB	1/2 Ht. 85ms starting at	\$438.
38MB	Full Ht. 40ms	\$788.
51MB	Full Ht. 40ms	\$928.

FLOPPY DRIVES

PC COMPATIBLE

PANASONIC 1/2 Ht. \$89.88

SANYO 1/2 Ht. \$79.00

APPLE II Compatible, inc. cable \$97.75

MODEMS

SmarTEAM 103/212A

1200/300 baud auto answer/auto dial, Hayes™(AT) compatible external modem. Monitor speaker, tone/pulse dialing, status indicator lights.

\$178.00

2 Year Warranty

ZOOM 1200

1200/300 baud, auto answer/auto dial, Hayes™(AT) compatible internal modem. Deamon dialing, monitor speaker, tone/pulse dialing, dual phone jacks, call progress tone detection. 2 Year Warranty. Made in USA.

\$199.88

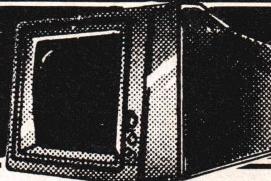
INCLUDING SOFTWARE!

MONITORS

TATUNG

The highest image quality, most reliable monitors we've tested.

1360 13" Hi-Res RGB 640 X 200	\$367
1370 13" Ultra Hi-Res RGB 720 X 400	\$447



BORLAND

REPEAT OF A SELLOUT!

SIDEKICK 1.5 c/p	\$27.75
SIDEKICK 1.5 nc/p	\$42.99
SIDEKICK MAC nc/p	\$42.99
TURBO PASCAL 3.0 nc/p	\$35.33
SUPERKEY 1.1 nc/p	\$35.33

TRAVELING SIDEKICK nc/p	\$39.97
TURBO LIGHTNING nc/p	\$55.00
REFLEX nc/p	\$55.00

FUJI

Certified Quality 5 1/4" FUJI Bulk Disks With or without box.

DS/DD in White Box

With sleeves, labels and w/p tabs. Priced per box of 10.

1-19	20-99	100-499	500-999	1000+	60-180	200-480	500-980	1000-4980	5000+
9.99	9.30	8.79	8.32	7.99	.92	.82	.76	.71	.67

DISKETTES

NEW LOWER PRICES!

DS/DD Bulk w/o box, sleeves, labels or w/p tabs

Price each, in increments of 20 only, poly bagged.

ACCESSORIES

PC POWER SUPPLIES

150 WATT	\$94
KEYBOARDS: IBM/XT™ and KEYTRONICS™ COMPATIBLE:	
5150 style	\$69
5151 style	\$89

REPLACEMENT HARD DISK DRIVES:

25MB 1/2ht. 85ms or better	\$348
25MB Full ht. 40ms	\$568
38MB Full ht. 40ms	\$678
51MB Full ht. 40ms	\$848

BOARD PRODUCTS:

Western Digital PC/AT type Hard/Floppy cont.	\$279
MULTITECH 2 Drive PC floppy controller	\$45
MULTITECH 4 drive PC floppy controller	\$58
AST Advantage 128K	\$384
MULTITECH color graphics board (1-2-3 comp.)	\$98
HERCULES graphics board	\$287

QUADRAM

Quadboard w/64K \$197

MULTITECH 384K mem. exp. board (empty) \$49

MODEMS:

NOVATION SMARTCAT+ 1200/300 baud, int. or ext. Auto ans/Auto dial, Hayes™(AT) comp. w/MITE \$308

HAYES 1200B w/Smartcom II \$349

NOVATION 2400 with MS-DOS or Macintosh software \$548

ZOOM IIE 300 baud AA/AD Hayes™ comp. W/software \$96.00

MONITORS:

AMDEK 710 RGB 720 X 400 \$486

PRINCETON MAX-12 Amber \$166

PRINCETON HX-12E RGB \$544

TATUNG 1222 12" Amber \$118

TAXAN 122 monochrome \$137

TAXAN 640 RGB 720 X 400 \$529

Everybody hates us but our customers.

6311-L Desoto Ave., WOODLAND HILLS, CA 91367

HOURS: 7:00AM-6:00PM Pacific Time

**IF YOU DON'T SEE IT, CALL!

We have virtually any product available at the best pricing. Space limits us to only a fraction of what we sell. Call us for a quote and delivery information. If we don't have it, we'll try to get it for you! All items priced and in stock at time of ad placement and subject to vendor changes and prior sale at time of ad publication.

Dealers: Call for quantity prices!

METRIC MINIMIZER

LISTING TWO (Listing continued, text begins on page 24)

```
DATA    data[] ;          /* dummy */  
int     npoints ;        /* dummy */  
int     user ;           /* dummy, reserved for each user! */  
{  
/*  
 * computes the function to be minimized  
 *  
 * this is the test function given in Cohen, page 280  
 * note that there are no constants for this fcn --  
 * the vector const is included for compatibility with other fcns  
 *  
 * note program calls function with user = 0  
 */  
  
double t1 , t2 , t3 , t4 ;  
  
t1 = (xx[1] - xx[0]*xx[0]) ;  
t2 = t1 * t1 ;  
t3 = (xx[0] * (1.0 - xx[1]) + 1.0) ;  
t4 = t3 * t3 ;  
return ( t2 + t4 ) ;  
}  
  
/********************************************/  
  
double sinus ( const , xx , data , npoints , user )  
double const[] ;           /* vector of constants for this fcn */  
double xx[] ;              /* parameter vector */  
DATA    data[] ;           /* data from file */  
int     npoints ;          /* number of data points */  
int     user ;             /* user parameter */  
{  
/*  
 * extract parameters for polynomial fit to sine function  
 */  
  
double t1 , t2 , t3 , t4 ;  
int     i;  
  
for ( i=0 , t4 = 0.0 ; i<npoints ; i++ )  
{  
    t1 = data[i].x * data[i].x ;  
    t2 = data[i].x *  
          (xx[0] + t1*(xx[1] + t1*(xx[2] + t1*xx[3])) ) ;  
    t3 = (data[i].y - t2)/data[i].y ;  
    t4 += t3 * t3 ;  
}  
  
return( t4 /= (double)(npoints-4) ) ;  
}  
  
/********************************************/  
  
double rosen ( const , xx , data , npoints , user )  
double const[] ;           /* vector of constants used by this fcn */  
double xx[] ;              /* the current values of the vector x */  
DATA    data[] ;           /* dummy */  
int     npoints ;          /* dummy */  
int     user ;             /* dummy, reserved for each user! */  
{  
/*  
 * Rosenbrock's test function  
 */  
  
double t1 , t2 , t3 ;  
int     l , m ;  
  
for ( l=0 , m=0 , t3 = 0.0 ; l<10 ; l += 2 , ++m )  
{  
    t1 = xx[l] ;  
    t2 = xx[l+1] ;  
    t3 += const[m]*(t2-t1*t1)*(t2-t1*t1) + (1.0-t1)*(1.0-t1) ;  
}  
return( t3 ) ;  
}
```

End Listing Two

LISTING THREE

```
/* MINIM.C  
*  
* The main minimization routine  
* (c) Copyright 1985, Billybob Software. All rights reserved.  
* This program may be reproduced for personal, non-profit use only.  
*/  
  
#include      "global.h"
```

(Continued on page 82)

DDJ BACK ISSUES

#73 Volume VII, Issue 11:

Wildcard UNIX Filenames—Tests for Pidgin—68000 Cross Assembler Listing, Part 2—Adding More BDOS Calls—The Perfect Hash—BASIC Memory Management—Benchmarks for CP/M-86 vs. MSDOS, and the 8087.

#78 Volume VIII, Issue 4:

RECLAIM Destroyed Directories—Binary Magic Numbers—8080 Fig-Forth Directory & File System—SAY! Forth Votrax Driver—TRS-80 8080 to Z80 Translator—Basic Disk I/O, Part II.

#79 Volume VIII, Issue 5:

The Augusta Compiler—A Fast Circle Routine—Enhancing the C Screen Editor—Shifts and Rotations on the Z80—The SCB, TSX, and TXS Instructions of the 6502 and 6800—MS-DOS vs. CP/M-86—Controlling MBASIC—The Buffered Keyboard—IBM PC Character Set Linker—Flip Utility for the IBM PC.

#80 Volume VIII, Issue 6:

Fast Divisibility Algorithms—B-Tree ISAM Concepts—CP/M BDOS and BIOS Calls for C—Serial Expansion in Forth—Fast Matrix Operations in Forth, Part I—Yes, You Can Trace Through BDOS—Julian Dates for Microcomputers—8088 Addressing Modes—8088 Line Generator—CP/M Plus.

#81 Volume VIII, Issue 7:

The Augusta Compiler, continued—RED: A Better Screen Editor, Part I—Anatomy of a Digital Vector and Curve Generator—Fast Matrix Operations in Forth, Part II—The AGGHHHI Program—MBOOT Revisited—CP/M Plus Feedback—MS-DOS Rebuttal—68000 Tools—Sizing Memory on the IBM PC.

#82 Volume VIII, Issue 8:

Serial-to Parallel: A Flexible Utility Box—McWORDER: A Tiny Text Editor—And Still More Fifth Generation Computers—Specialist Symbols and I/O Benchmarks for CP/M Plus—CP/M Plus Memory Management—Zero Length File Test—PAUSEIF, QUITIF, and now SKIPIF—ACTxx Cross Assemblers.

#83 Volume VIII, Issue 9:

FORTH ISSUE: Forth and the Motorola 68000—Nondeterministic Control Words in Forth—A68000 Forth Assembler—GO in Forth—Precompiled Forth Modules—Signed Integer Division—Some Forth Coding Standards—The Forth Sort.

#84 Volume VIII, Issue 10:

Unix to CP/M Floppy Disk File Conversion—A Small-C Help Facility—Attaching a Winchester Hard Disk to the S-100 Bus—Using Epson Bit-Plot Graphics—8086/88 Function Macros—Auto Disk Format Selection—CP/M Plus Device Tables.

#85 Volume VIII, Issue 11:

A Kernel for the MC68000—A DML Parser—Towards a More Writable Forth Syntax—Simple Graphics for Printer—Floating-Point Benchmarks.

#86 Volume VIII, Issue 12:

Faster Circles for Apples—Cursor Control for Dumb Terminals—Dysan's Digital Diagnostic Diskette—Interfacing a Hard Disk Within a CP/M Environment—The New MS-DOS EXEC Function.

#87 Volume IX, Issue 1:

A structured Preprocessor for MBASIC—A Simple Window Package—Forth to PC-DOS Interface—Sorted Diskette Directory Listing for the IBM PC—Emulate WordStar on TOPS-20—More on optimizing compilers—The PIP mystery device contest.

#88 Volume IX, Issue 2:

Telecommunications Issue: Micro To Mainframe Connection—Communications Protocols—Unix to Unix Network Utilities—VPC: A Virtual Personal Computer for Networks—PABX on the Personal Computer—BASIC Language Telecommunications Programming—U.S. Robotics S-100 Card Modem.

#89 Volume IX, Issue 3:

RSA: A Public Key Cryptography System, Part I—Introduction to PL/C: Programming Language for Compilers—Program Design Using Pseudocode—More on Binary Magic Numbers—How fast is CP/M Plus?—CP/M 2.2 BIOS Function: SELDSK—The results of the Floating-Point benchmark.

TO ORDER:

Return this coupon with payment to: Dr. Dobb's Journal, 501 Galveston Dr., Redwood City, CA 94063

Please send me the issue(s) circled: 71 72 73 78 79 80
81 82 83 84 85 86 87 88 89 90 91 92 95 96
97 104 105 106 107 108 109 110 111 112

Price: 1 issue—\$5.00, 2-5 issues—\$4.50 each, 6 or more—\$4.00 each.
(There is a \$10.00 minimum for charge orders.)

I have read the postal instructions and understand that I will not receive my order unless I have sent the correct payment amount.

Please charge my: Visa M/C Amer. Exp.

Card No. _____ Exp. Date _____

Signature _____

#90 Volume IX, Issue 4:

Optimizing Strings in C—Expert Systems and the Weather—RSA: A Public Key Cryptography System, Part II—Several items on CP/M Plus, CP/M 2.2 Compatibility—BDOS Function 10: Vastly improved—More on MS-DOS EXEC Function—Low-Level Input-Output in C.

#91 Volume IX, Issue 5:

Introduction to Modula-2 for Pascal Programmers—Converting Fig-Forth to Forth-83—Sixth Generation Computers—A New Library for Small-C—Solutions to Quirks in dBASE II.

#92 Volume IX, Issue 6:

CP/M on the Commodore 64—dBASE II Programming Techniques—First Chinese Forth: A Double-Headed Approach—cc-A Driver for a Small-C Programming System—A New Library for Small-C (Part II)—Comments on Sixth Generation Computers—Review of Turbo Pascal.

#95 Volume IX, Issue 9:

Forth Special Issue—File Maintenance in Forth—Forth and the Fast Fourier Transform—Computing with Streams—A Forth Native-Code Cross Compiler for the MC68000—The FVG Standard Floating-Point Extension—CP/M Plus: Interbank Memory Moves Without DMA—Ways to make C more powerful and flexible.

#96 Volume IX, Issue 10:

More dBASE II Programming Techniques—Simple Calculations with Complex Numbers—GREP.C: A Unix-like Generalized Regular Expression Parser—An optimization scheme for compilers, MSDOS 2.0 Filters, Sizing RAM under MSDOS, Two programming systems illustrating Runge-Kutta integration.

#97 Volume IX, Issue 11:

Adding Primitive I/O Functions to muLISP—Program Monitor Package: Using Interrupts to Instrument Applications—CP/M 2.2 Goes PUBLIC—A Guide to Resources for the C Programmer—RESORT.

#104 Volume X, Issue 6:

Information Age Issues—Modems: 2400 Bit/Sec and Beyond—C UART Controller—Christensen Protocols in C.

#105 Volume X, Issue 7:

Build a Custom PC or Clone—The Ultimate Parallel Print Spooler—Designing a Real-Time Clock for the S-100 Bus.

#106 Volume X, Issue 8:

C Compilers for MSDOS—A Peephole Optimizer for Assembly Language Source Code—Small C Update—Asynchronous Protocols.

#107 Volume X, Issue 9:

Parallel Pattern Matching and Egrep—Bose-Nelson Sort—Two TEx Implementations for the IBM PC.

#108 Volume X, Issue 10:

Special Forth Issue—A Threaded-Code Microprocessor Bursts Forth—Design a Forth Target Compiler.

#109 Volume X, Issue 11:

Modula-2 vs. Pascal for Microcomputers: An Update—Bit Manipulation in Modula-2.

#110 Volume X, Issue 12:

GEM, Topview and Windows as Programming Environments—Operating System Extensions for CP/M-68K, ProDOS, CP/M—Converting to DOS 3.0.

#111 Volume XI, Issue 1:

Bringing Up a 68K Machine from Scratch—PL/68K: Is it C or is it Assembler—An 8080 Simulator for the 68K—DOS Shell Notes on 80286 Big DOS.

#112 Volume XI, Issue 2:

Structured code in Pascal, Modula-2, Ada—the Great CRC Mystery: Solved—Where to find public domain Ada software—Data Abstraction with Modula-2—A shell for MS DOS.

I enclose \$ _____ (U.S. check or money order).

Outside the U.S., add \$.50 per issue.

Name _____

Address _____

City _____

State _____ Zip _____

Outside U.S., add \$.50 per issue. Price includes shipment by second class or foreign surface mail. Within U.S., allow 9 weeks for delivery. For U.P.S. shipment, add \$1.00 for 1-2 issues and \$.50 per issue thereafter—NO P.O. BOX. Airmail rates: Canada—add \$1.75 per issue; other foreign add \$3.00 per issue. Please provide a street address rather than a P.O. Box.

METRIC MINIMIZER

LISTING THREE (Listing continued, text begins on page 24)

```
*****  
minim ( n , fun , xextern , xstep , xlim , g0 , itermin , iterlim , epsilon ,  
gmin , iters , debug , nreset , data , npoints , const , method )  
  
int n ; /* number of parameters in the fit */  
double (*fun)() ; /* pointer to the function to minimize */  
double xextern[] ; /* vector containing the starting values of  
the parameters; returns values at minimum */  
double xstep[] ; /* step sizes on parameters for deriv calcs */  
BOUND xlim[] ; /* limits on the vector x */  
double g0 ; /* expected value of the minimum of the fcn */  
int itermin ; /* minimum number of iterations */  
int iterlim ; /* limit on the number of iterations */  
double epsilon[] ; /* iteration control parameters */  
double *gmin ; /* value of minimum for returned vector x */  
int *iters ; /* number of iterations to converge */  
int debug ; /* flag = 0 for no print, >0 for debug print */  
int nreset ; /* reset limit for the y matrix */  
DATA data[] ; /* the data */  
int npoints ; /* number of data points */  
double const[] ; /* vector of constants required by fcn */  
int method ; /* update method for the matrix y */  
{  
/*  
* variable metric minimization algorithm  
* Reference (using DFP method) is: Numerical Analysis  
* A.M. Cohen et. al.  
* Halstead Press  
* John Wiley and Sons  
* 1973  
* pages 276-280  
* See also original papers by Fletcher and Powell  
*/  
  
int i , j , rc ;  
static int ycount ; /* counter for resetting the Y matrix */  
static double y[MATMAX] ; /* the Y matrix */  
static double a[MATMAX] ; /* the A and B matrices are calculated to */  
static double b[MATMAX] ; /* update the Y matrix at each iteration */  
static double xintern[VECMAX];/* parameters in internal coordinates */  
static double zint[VECMAX] ; /* gradients in internal coordinates */  
static double znewi[VECMAX] ; /* same for the next iteration */  
static double sigma[VECMAX] ; /* changes vector */  
static double s[VECMAX] ; /* changes direction vector */  
static double xi[VECMAX] ; /* change in gradient vector between */  
/* iterations */  
static double xold[VECMAX] ; /* holds previous values of x */  
static double alpha ; /* */  
static double g ; /* value of fcn for a given vector x */  
static double gnew ; /* likewise for the next iteration */  
double dot() ;  
  
/*  
* get started  
*/  
  
g = (*fun)( const , xextern , data , npoints , 0 ) ;  
  
gozinta( n , xintern , xextern , xlim ) ;  
derivs ( n , fun , xintern , xstep , xlim , zint , data ,  
npoints , const , debug );  
/*  
* main loop  
*/  
  
for ( i = 1 , ycount = nreset ; i <= iterlim ; ++i , ++ycount )  
{  
if (ycount == nreset)  
{  
    reset( n , y );  
    ycount = 0 ;  
}  
if (debug)  
{  
    printf("\t\t+++++++\n");  
    printf("\t\t Begin iteration %d \n", i );  
}
```

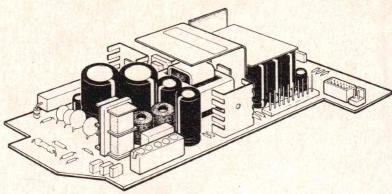
(Continued on page 84)

DIGITAL RESEARCH COMPUTERS

(214) 225-2309

Switching Power Supply!

- + 5VDC - 8 Amps
- +12VDC - 5 Amps
- 12VDC - 1 Amp
- (81 WATTS MAX)



\$29.95
ea.

4 FOR \$99

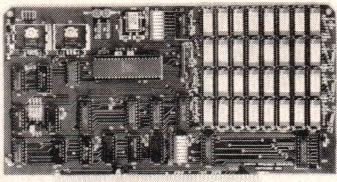
ADD \$2
EA. UPS

BRAND NEW UNITS, MFG. BY BOSCHERT FOR HEWLETT PACKARD! PERFECT FOR SMALL COMPUTER AND DISK DRIVE APPLICATIONS #XL81-5630. INPUT 110V/220V, 50/60 HZ. NOMINAL OUTPUTS: +5VDC @ 8A, +12VDC @ 5A, -12VDC @ 1A. TOTAL MAX OUTPUT MUST BE LIMITED TO 81 WATTS TOTAL! (WITH PIN OUT SHEET.) ORIGINAL FACTORY BOXED.

256K S-100 SOLID STATE DISK SIMULATOR!

WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

PRICE CUT!



BLANK PCB
(WITH CP/M® 2.2
PATCHES AND INSTALL
PROGRAM ON DISKETTE)
\$69.95
(8203-1 INTEL \$29.95)

\$149.00
(ADD \$50 FOR A&T)
#LS-100 (FULL 256K KIT)

FEATURES:

- * 256K on board, using + 5V 64K DRAMS.
- * Uses new Intel 8203-1 LSI Memory Controller.
- * Requires only 4 Dip Switch Selectable I/O Ports.
- * Runs on 8080 or Z80 S100 machines.
- * Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
- * Provisions for Battery back-up.
- * Software to mate the LS-100 to your CP/M® 2.2 DOS is supplied.
- * The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
- * Compare our price! You could pay up to 3 times as much for similar boards.

THE NEW ZRT-80

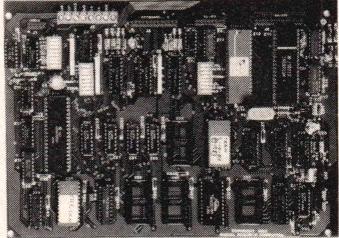
CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

FEATURES:

- * Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
- * RS232 at 16 BAUD Rates from 75 to 19,200.
- * 24 x 80 standard format (60 Hz).
- * Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
- * Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
- * Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
- * 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
- * Composite or Split Video.
- * Any polarity of video or sync.
- * Inverse Video Capability.
- * Small Size: 6.5 x 9 inches.
- * Upper & lower case with descenders.
- * 7 x 9 Character Matrix.
- * Requires Par. ASCII keyboard.

FOR 8 IN. SOURCE DISK
(CP/M® COMPATIBLE)
ADD \$10



\$89.95
#ZRT-80
(COMPLETE KIT, 2K VIDEO RAM)

BLANK PCB WITH 2716
CHAR. ROM. 2732 MON. ROM

\$49.95

SOURCE DISKETTE - ADD \$10
SET OF 2 CRYSTALS - ADD \$7.50

Digital Research Computers

P.O. BOX 381450 • DUNCANVILLE, TX 75138 • (214) 225-2309

64K S100 STATIC RAM **\$119.00** KIT

LOW POWER!

150 NS ADD \$10

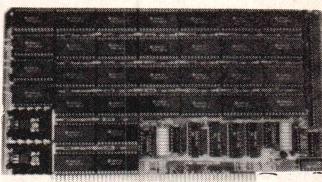
BLANK PC BOARD
WITH DOCUMENTATION
\$49.95

SUPPORT ICs + CAPS
\$17.50

FULL SOCKET SET
\$14.50

FULLY SUPPORTS THE
NEW IEEE 696 S100
STANDARD
(AS PROPOSED)
FOR 56K KIT \$105

ASSEMBLED AND
TESTED ADD \$50



FEATURES: PRICE CUT!

- * Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- * Fully supports IEEE 696 24 BIT Extended Addressing.
- * 64K draws only approximately 500 MA.
- * 200 NS RAMs are standard. (TOSHIBA makes TMM 2016 as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- * SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
- * 2716 EPROMs may be installed in any of top 48K.
- * Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
- * Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
- * BOARD may be partially populated as 56K.

PANASONIC

Green Screen - Video Monitors

25 MHZ. TYPICAL BANDWIDTH!!!

Brand New In The Box! 9-Inch Screen

#K-904B1 (Chassis #Y08A) Open Frame Style

\$29.95

EA.

GROUP SPECIAL:
4 for \$99.00
WITH DATA & SCHEMATIC

(USA SHIPPING: \$3. PER UNIT. CANADA: \$7. PER UNIT)

COMPUTER MANUFACTURER'S EXCESS. STILL IN ORIGINAL PANASONIC BOXES. THE CRT TUBE ALONE WOULD COST MORE THAN OUR PRICE FOR THE COMPLETE UNIT. FOR SPLIT VIDEO (TTL INPUTS) OPERATION, NOT COMPOSITE VIDEO. OPERATES FROM 12VDC AT 1 AMP. VERTICAL INPUT IS 49 TO 61 HZ. HORIZONTAL INPUT: 15,750 HZ ± 500 HZ. RESOLUTION IS 800 LINES AT CENTER 650 LINES AT CORNERS.

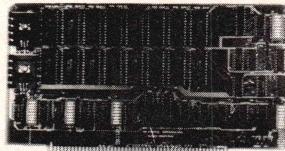
32K S100 EPROM/STATIC RAM

NEW!

FOUR FUNCTION BOARD!

NEW!

EPROM II
FULL
EPROM KIT
\$69.95
A&T EPROM
ADD \$35.00



BLANK
PC BOARD
WITH DATA
\$39.95

SUPPORT
IC'S
PLUS CAPS
\$16

FULL
SOCKET SET
\$15

We took our very popular 32K S100 EPROM Card and added additional logic to create a more versatile EPROM/RAM Board.

FEATURES:

- * This one board can be used in any one of four ways:
 - As a 32K 2716 EPROM Board
 - As a 32K 2732 EPROM Board (Using Every Other Socket)
 - As a mixed 32K 2716 EPROM/2K x 8 RAM Board
 - As a 32K Static RAM Board
- * Uses New 2K x 8 (TMM2016 or HM6116) RAM's
- * Fully Supports IEEE 696 Bus Standard (As Proposed)
- * Supports 24 Bit Extended Addressing
- * 200 NS (FAST!) RAM's are standard on the RAM Kit
- * Supports both Cromemco and North Star Bank Select
- * Supports Phantom
- * On Board wait State Generator
- * Every 2K Block may be disabled
- * Addressed as two separate 16K Blocks on any 64K Boundary
- * Perfect for MP/M® Systems
- * RAM Kit is very low power (300 MA typical)

32K STATIC RAM KIT — \$99.95

For RAM Kit A&T — Add \$40

TERMS: Add \$3.00 postage. Orders under \$15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over \$50 add 85¢ for insurance.

METRIC MINIMIZER

LISTING THREE (Listing continued, text begins on page 24)

```
printf("\t\t+++++\n");
printf("external parameter values are:\n");
vout( n , xextern );
printf("internal parameter values are:\n");
vout( n , xintern );
printf("internal gradient vector is:\n");
vout( n , zint );
printf("approximate inverse hessian matrix is:\n");
mout( n , y );
printf("the current minimum is ---%11.4e<---\n", g );
}

matvec ( n , y , zint , s ) ;
for (j=0 ; j<n ; ++j)
{
    xold[j] = xintern[j];
    s[j] *= -1.0;
}

if (debug>1)
{
    printf("vector s is:\n");    vout( n , s );
}

/*
 * do line search and quit if it fails
 * line search was successful if getalpha returns OK ( zero )
 */

if ( rc = getalpha ( n , fun , xintern , xstep , xlim , zint ,
                     s , g , g0 , &alpha , debug , data , npoints , const ) )
    break;

/*
 * update parameters
 */

for (j=0 ; j<n ; ++j)
{
    sigma[j] = alpha * s[j];
    xintern[j] += sigma[j];
}
if (debug>1)
{
    printf("vector sigma is:\n");    vout( n , sigma );
}

/*
 * get values for next iteration
 */

gozouta ( n , xintern , xextern , xlim );
gnew = (*fun)( const , xextern , data , npoints , 0 );

derivs (n , fun , xintern , xstep , xlim , znewi ,
        data , npoints , const , debug ) ;

for (j=0 ; j<n ; ++j)
    xi[j] = znewi[j] - zint[j];

if (debug>1)
{
    printf("orthogonality satisfied to %11.4e\n",
           dot( n , znewi , sigma ) );
    printf("vector xi is:\n");
    vout( n , xi );
}

/*
 * decide whether to continue or not
 * continue if decide returns OK ( zero )
 */

if ( rc = decide( g , gnew , n , znewi , sigma , y ,
                  epsilon , iterm , i , debug ) )
    break;

/*
 * update all other quantities for the next pass

```

(Continued on page 86)

At GTE Government Systems, Our People Can Afford to be Entrepreneurs.

GTE Government Systems takes you beyond the entrepreneurial spirit. We back up your initiative and enthusiasm with the resources of one of America's largest corporations. Consequently our people not only feel free to strive for innovative solutions, they also have the opportunity to see their ideas at work in the real world.

Put your talents to work at any one of the many diverse GTE Government Systems areas of specialization. Accept the challenges presented by artificial intelligence, signal processing, voice technology, lasers, and VLSI. Work on key defense programs employing Ada and other new languages. Participate in design and engineering projects which cross the borders of the impossible.

Innovation, initiative, and independence are rewarded with full GTE support. Reach your potential working with the best equipment, the latest methodologies, and the most talented professionals in your field. We provide the tools, the room, and the environment . . . the rest is up to you.

Some of the current career opportunities at GTE Government Systems include:

SYSTEMS ENGINEERING

We are seeking Systems Engineers and Technical Managers for:

- High-speed digital communication systems, TDMA/TDM/PCM
- Spread spectrum communication systems
- Mission planning and control systems
- EW/ESM/ECM systems
- AI/Expert systems
- Direction-finding subsystems
- Receiving and Processing subsystems
- Real-time processing
- Signal analysis
- RF tracking systems
- Over-the-horizon radar

DIGITAL HARDWARE ENGINEERING

- VHSIC/VLSI design
- Digital design
- Digital signal processing
- Microprocessor hardware/software
- Communications signal processing

RF HARDWARE ENGINEERING

- State-of-the-art microwave component design
- High power component design for RF transmitter



- Leadership opportunities targeting technology developments
- System applications requiring creative hardware design contributions

MECHANICAL ENGINEERING

- Electro-mechanical packaging and miniaturization
- EMI/TEMPEST packaging
- Thermal/structure design and analysis
- Project Management

MIL-SPEC experience required.

SOFTWARE ENGINEERING

Opportunities exist for skilled software engineers across a full range of defense system applications using development systems such as DEC/VAX 8600 and MicroVAX II, PDP 11 series, and the Hewlett-Packard 1000. Microprocessor applications are directed to Intel 8085/8086, TI 32020, and Motorola 68000 processors. We are seeking senior people in the following areas:

● ADA

Use the DEC/VAX and MicroVAX II in our advanced development facility to build systems requiring:
—Real-time analysis and control

- Advanced MMI: multiple screen, color graphics
- Relational DBMS
- Networking on ETHERNET/VAX clusters
- Large system development (300,000 lines of source code)
- Microprocessor applications (Motorola 68020)

● FIXED AND AIR-SEA-LAND MOBILE SYSTEMS

- Interdisciplinary software/firmware/hardware teams build systems from high-level design through final test and integration using skills in:
- Microprocessor applications
 - Real-time analysis and control
 - Test-bed development and analysis
 - Mapping and graphics displays
 - Man-machine interfacing
 - Concurrent processing
 - Software architecture design
 - Data base design and applications
 - AI/Expert systems

THE BAY AREA

The San Francisco Bay area is one of the world's most attractive locations. Here, geographic diversity is enhanced by fine climate, cultural richness and an abundance of recreational opportunities. This energetic and dynamic environment also provides a multitude of educational opportunities—many of America's most outstanding universities are in close proximity.

GTE Government Systems offers you a competitive compensation package, a professional work environment, and complete benefits which include educational assistance, a stock purchase plan, a tax-deferred savings plan, and much more. Please send your resume in complete confidence to:

GTE Government Systems Corporation Western Division

Dept. CC 275
P.O. Box 7188
100 Ferguson Drive
Mountain View, CA 94039

An equal opportunity employer. U.S. citizenship is required.



Government Systems

METRIC MINIMIZER

LISTING THREE (Listing continued, text begins on page 24)

End Listing Three

Listings to be continued next month

The Conference That Takes A Hard Look At The Business Side Of Software.

The Software Business Conference April 2 & 3, Los Angeles, California

This prestigious conference, held in conjunction with COMDEX/Winter, will provide a forum for developers, publishers and resellers of software to discuss common interests, challenges, and solutions.

The Software Business Conference will also create an environment in which to discuss and establish new business relationships.

By bringing the three main bodies of the software industry together at one major conference, the Software Business Conference proposes to raise the visibility of software issues worldwide, and to assist in bringing high quality products to the market.

To ensure that the Conference meets its goals and is relevant to your interests, these and other noted distinguished industry leaders are helping to shape this major event: Joel Berez, President, Infocom, Inc.; Daniel Bricklin, President, Software Garden, Inc.; Seymour Rubenstein, President, MicroPro International Corp.; Gary Kildall, President, ActiVenture Corp.

Software Business Conference Schedule

Plenary Sessions

A Marriage of Objectives; Keeping Pace with a Changing PC Environment.

Marketing Sessions

What's Selling? The Software Marketplace; Who's Buying? Applications and Users.

Presented by



THE
INTERFACE
GROUP, Inc.

The world's leading producer of computer and communications conferences and expositions.

300 First Avenue, Needham, MA 02194

Circle no. 260 on reader service card.

General Sessions

Managing the Successful Software Company; Software Development Technical Strategies; Software Development Management Strategies; Exploiting the New Software Product; Supporting the New Software Product; The Great Code Rush: Prospecting for New Packages

Workshops in Software Entrepreneurship

Workshop in Creative Financing; Workshop in Creative Marketing and Selling

To be part of this exciting business and learning experience, send today for complete registration information.

— — — — —
Yes, I Want To Take A Hard Look At Software, Too.
Send Me Detailed Information.

Name _____

Address _____

City/State/Zip _____

Phone _____

I'm a Developer Publisher Reseller Other

Mail to: Software Business Conference
300 First Avenue
Needham, MA 02194

©Copyright 1986

CONCURRENCY

LISTING ONE (Text begins on page 36)

```

{task #1: keyboard -> serial out
task #2: serial in -> video out
control-C will abort program}
program main;
const   TASKS=2;
        STACKSIZE=70;
{next 7 constants are needed for the Kaypro}
KDATA=5;
KSTAT=7;
BAUDP=0;
SDATA=4;
SSTAT=6;
RMASK=1;
TMASK=4;

CC=3;
type stack = array[0..STACKSIZE] of integer;
tasknum = -1..TASKS;
var sp0,sp1,sp2: integer; {when zero, task not initialized}
oldn: tasknum;
nextn: tasknum;
Procedure defer; forward;
procedure exit;
begin
  writeln('TASK #',oldn,' terminated.');
  oldn:=-1;
  defer;
end;
function keyin:byte;
begin
  repeat
    defer;
    until (RMASK = (RMASK and port[KSTAT]));
    keyin:= port[KDATA];
  end;
procedure videotout (b:byte);
begin
  bdos(6,b);
end;
function serin: byte;
begin
  repeat
    defer;
    until (RMASK = (RMASK and port[SSTAT]));
    serin:= port[SDATA];
  end;
procedure serout (b:byte);
begin
  repeat
    defer;
    until (TMASK = (TMASK and port[SSTAT]));
    port[SDATA]:=b;
  end;
Procedure task1;
var mystack: stack;

```

```

key: byte;
begin
stackptr:=addr(mystack[STACKSIZE]);
repeat
  key:=keyin;
  if key=CC then exit
  else serout(key);
until false;{forever}
exit;
end;
Procedure task2;
var mystack: stack;
begin
stackptr:=addr(mystack[STACKSIZE]);
repeat
  videotout(serin);
until false{forever};
exit;
end;
procedure initall;
var i: integer;
begin
  sp1:=0;
  sp2:=0;
  oldn:=0;
  {initialize Kaypro's SIO}
  port[BAUDP]:=14; {9600 Baud}
  port[SSTAT]:=24;
  port[SSTAT]:=4;
  port[SSTAT]:=68;
  port[SSTAT]:=1;
  port[SSTAT]:=0;
  port[SSTAT]:=3;
  port[SSTAT]:=193;
  port[SSTAT]:=5;
  port[SSTAT]:=234;
end;
Procedure schedule;
begin
  if oldn=TASKS then nextn:=1
  else nextn:=oldn+1;
end;
procedure defer;
var sp: integer;
begin
  case oldn of
    0: sp0:=stackptr;
    1: sp1:=stackptr;
    2: sp2:=stackptr;
  end{case};
  schedule;
  oldn:=nextn;
  case nextn of
    0: sp:=sp0;
    1: sp:=sp1;

```

(Continued on page 90)

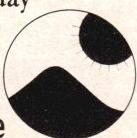
TRUE MULTI-TASKING!

TASKMASTER is high tech, available now, and it works with virtually all DOS software. Give Lotus, Sidekick, Multimate or most any DOS program the advantages of real multi-tasking. It's simple to use, compatible, bulletproof and most of all, it won't slow you down. That's because TASKMASTER only shares your computer when YOU want it shared. At other times, your visible program runs at full speed, waiting for you to easily switch from program to program at the touch of a key. Compatible with most DOS computers including the IBM PC/XT/AT/Jr. series, you can order TASKMASTER today for only \$69.95 + 5.00 Shipping and Handling, VISA and Mastercard.

ORDER LINE
(206)367-0650

Taskmaster trademark Sunny Hill Software.
Lotus trademark Lotus Development Corp.
Sidekick trademark Borland Int'l.
Multimate trademark Ashton Tate.

**Sunny Hill
Software**



13732 Midvale North Suite 206
Seattle, Washington 98133

Circle no. 172 on reader service card.

BOOKSHELF™

Series 100

**Fast, compact, high quality,
easy-to-use CP/M system**



Priced from
\$895.00

**10MB System
Only \$1645.00**

Features

- Ready-to-use professional CP/M computer system
- Works with any RS232C ASCII terminal (not included)
- Network available
- Compact 7.3 x 6.5 x 10.5 inches, 12.5 pounds, all-metal construction
- Powerful and Versatile:
 - Based on Little Board/PLUS single-board computer
 - Two RS232 serial ports
 - One Centronics printer port
 - One or two 400 or 800 KB floppy drives
 - 10-MB internal hard disk drive option

- Comprehensive Software Included:
 - Enhanced CP/M operating system with ZCP3
 - Word processing, spreadsheet, relational database, spelling checker, and data encrypt/decrypt (T/MAKER III)
 - Operator-friendly shells; Menu, Friendly™
 - Read/write and format dozens of floppy formats (IBM PC-DOS, KAYPRO, OSBORNE, MORROW...)
 - Menu-based system customization
- Expandable:
 - Floppy expansion to four drives
 - Hard disk expansion to 60 megabytes
 - SCSI/PLUS™ multi-master I/O expansion bus

AMPRO
COMPUTERS INCORPORATED

67 East Evelyn Ave. • Mountain View, CA 94041 • (415) 962-0230 • TELEX 4940302

Circle no. 158 on reader service card.

T/MAKER III is a trademark of T/Maker Company.
IBM is a registered trademark of International Business Machines.
Z80A is a registered trademark of Zilog, Inc.
CP/M is a registered trademark of Digital Research.

I Q C L I S P

THE CLOSEST THING TO COMMON LISP AVAILABLE FOR YOUR PC

RICH SET OF DATA TYPES

Bignums, for high precision arithmetic
8087 support, for fast floating point
Arrays, for multidimensional data
Streams, for device-independent i/o
Packages, for partitioning large systems
Characters, strings, bit-arrays

FULL SET OF CONTROL PRIMITIVES

flet, labels, macrolet, for local functions
if, when, unless, case, cond, for conditionals
Keyword parameters, for flexibility
Multiple-valued functions, for clarity
Flavors, for object-oriented programming
Stacks, for coroutining
Closures, for encapsulation

LARGE COMPLEMENT OF FUNCTIONS

Mappers, for functional programming
format, for output control
sort, for user-specified predicates
Transcendental floating point functions
String handling functions
Over 400 functions altogether

APPLICATION SUPPORT

Save and restore full environments
User-specified initializations
Assembly language interface

HARDWARE REQUIREMENTS

8088 or 8086 CPU, MSDOS Operating System
390K RAM or more

IQCLISP

5 1/4" diskettes
and manual \$300.00

Foreign orders add \$30.00 for airmail.
U.S. Shipping included for prepaid orders.

fq Integral Quality

P.O. Box 31970
Seattle, Washington 98103
(206) 527-2918

Washington State residents add sales tax.
VISA and MASTERCARD accepted.

EXTENDABILITY

defstruct, to add data types
Macros, to add control constructs
Read macros, to extend the input syntax
Extendable arithmetic system
Customizable window system

DEBUGGING SUPPORT

step, for single-stepping
trace, for monitoring
break, for probing
inspect, for exploring
Flexible error recovery
Customizable pretty-printer

MSDOS INTERFACE

Random access files
Hierarchical directory access
MSDOS calls

DOCUMENTATION

On-line documentation of functions
apropos
300-page indexed reference manual

THE HAMMER

Software Tools in C

"I have already saved weeks of coding . . . thank you for providing such a useful tool . . ." - G.T.

Let The HAMMER Library of over 150 routines save you valuable development time and effort:

LIBRARY FUNCTIONS

- Multi-level 123-like MENUS
- DATA ENTRY
 - MULTI-FIELD mode for Full-Screen data entry
 - Single-Field mode for individual fields
 - Data Verification
 - Full Editing within each field
 - Strings, dates, and fixed decimal numbers
 - "Option" fields force user to pick from a given set
- SCREEN MANAGEMENT
 - cursor positioning
 - full attribute control
 - Date/time/string conversions
 - BIOS access/pattern matching/and more

UTILITIES

- HARC - complete Source File Archiver
- HAMCC - compile designated source modules residing **WITHIN** an archive file under any of the supported compilers and optionally place resulting object modules in a library.

SUPPORTED C COMPILERS:

Microsoft C 3.00 • CI-C86 • Mark Williams C86

DeSmet C • Lattice

INCLUDES source code and manual

\$195 plus shipping

VISA/MC accepted

O.E.S. SYSTEMS

1906 Brushcliff Rd. • Pittsburgh, PA 15221 • 412/243-7365

Looking for the right tool for the job?

REACH FOR THE HAMMER

Circle no. 137 on reader service card.

CONCURRENCY

LISTING ONE (Listing continued, text begins on page 36)

```

2: sp1:=sp2;
end{case};
if sp1>0 {initialized}
then begin
  stackptr:=sp1;
  end
else {not initialized}
begin
  writeln('Starting task #',nextn);
  case nextn of
  1: task1;
  2: task2;
  end{case};
  end;
end{defer};
begin{main}
initial;
writeln('Multitasking version of simple terminal program');
writeln('Control-C will terminate it');
writeln;
defer;
writeln('Main: done');
end.

```

End Listing One

LISTING TWO

```

(task #1: keyboard -> fifo1
task #2: fifo1 -> filter -> fifo2
task #3: fifo2 -> slow display )
program main;
const  TASKS=3;
STACKSIZE=20;
NFIFOS=2;{#1 is for input and #2 for output}
PRATE=300;{SLOWS the display function}
{the following three constants are for the Kaypro Computer}
KDATA=5; KSTAT=7; RMASK=1;
CR=13;
LF=10;
CC=3;
BS=8;
RUB=127;
SPACE=32;
CQ=17;{XON}
CS=19;{XOFF}
type stack = array[0..STACKSIZE] of integer;
fifo = record
  buf: array[0..255] of byte;
  inptr: byte;
  outptr: byte;
  flow: boolean;{for flow control}
end;
fifo1 = 1..NFIFOS;
tasknum = -1.. TASKS;
var  sp0,sp1,sp2,sp3: integer;{when zero,task not initialized}
oldn: tasknum;
nextn: tasknum;
fifos: array[1..NFIFOS] of fifo;
Procedure defer; forward;
function occupancy(p: fifo):byte;
begin with fifos[p] do
  occupancy:= inptr-outptr;
end;
function vacancy(p: fifo): byte;
begin with fifos[p] do
  vacancy:=outptr-inptr-1;
end;
function dequeue1: byte;
begin with fifos[1] do
  begin
    while (occupancy(1)=0) or not flow
      do defer;
    dequeue1:= buf[outptr];
    outptr:=outptr+1;
  end;
end;
function dequeue2: byte;
begin with fifos[2] do
  begin
    while (occupancy(2)=0) or not flow
      do defer;
    dequeue2:= buf[outptr];
    outptr:=outptr+1;
  end;
end;
procedure exit;

```

```

begin
writeln('JOB #',oldn,' terminated.');
oldn:=-1;
defer;
end;
procedure enqueue1(b:byte);
begin with fifos[1] do
  begin
    buf[inptr]:=b;
    while vacancy(1)=0 do
      defer;{hang while full}
    inptr:=inptr+1;
  end;
end;
procedure enqueue2(b:byte);
begin with fifos[2] do
  begin
    buf[inptr]:=b;
    while vacancy(2)=0 do
      defer;{hang while full}
    inptr:=inptr+1;
  end;
end;
function keyin:byte;
begin
  repeat until (RMASK = (RMASK and port[KSTAT]));
  keyin:= port[KDATA];
end;
procedure vout(b:byte);
begin
  bdos(6,b);
end;
Procedure print;{task#3}
var mystack: stack;
i: integer;
begin
  stackptr:=addr(mystack[STACKSIZE]);
  i:=0;
  {initialize fifo#2}
  with fifos[2] do
    begin
      outptr:=0;
      inptr:=0;
      flow:=true;
    end;
  repeat
    i:=i+1;
    if i=PRATE then
      begin
        i:=0;
        vout(dequeue2);
      end
    else
      defer;
  until false;{forever}
  exit;
end;
Procedure keyboard;{task #1}
var mystack: stack;
cb: byte;
begin
  stackptr:=addr(mystack[STACKSIZE]);
  {initialize fifo #1}
  with fifos[1] do
    begin
      inptr:=0;
      outptr:=0;
      flow:=true;
    end;
  repeat
    if (1 = (1 and port[KSTAT])) then
      begin
        cb:= port[KDATA];
        enqueue1(cb);
      end
    else defer;
  until false{forever};
  exit;
end;
Procedure filter;{task #2}
var mystack: stack;
b: byte;
begin
  stackptr:=addr(mystack[STACKSIZE]);
  repeat
    b:=dequeue1;
    case b of
    CR: begin

```

(Continued on page 93)

Parallel Programming for "C"

TEAMWORK

A Concurrent Programming Toolkit

Teamwork is a "C" program library which allows you to write your programs as a set of cooperating concurrent tasks. Very useful for simulation, real-time applications, and experimentation with parallel programming.

FEATURES

- Supports a very large number of tasks (typically more than 50) limited only by available memory. Low overhead per task results in very fast context switching.
- Provides a full set of inter-task communication (ITC) facilities, including locks, semaphores, blocking queues, and UNIX*-style signals. Also has building blocks for constructing your own ITC facilities.
- Handles interrupts and integrates them into task scheduling. Supply your own interrupt handlers or block tasks on interrupts.
- Lets you trace task switches and inter-task communication.
- Comes with complete documentation including a user's manual and reference manual of commands.

Teamwork is available for the following systems:

Hardware	Operating System	Price
IBM PC, XT, AT	PC-DOS 2.0 or later	\$ 65
IBM PC AT	XENIX*	\$ 85
DEC VAX*	UNIX 4.2BSD	\$195

PC-DOS version is compatible with DeSmet, Lattice, and Microsoft C compilers.

Please specify hardware and operating system when ordering. Send check or money order to:



Block Island Technologies
Innovative Computer Software

13563 NW Cornell Road, Suite 230, Portland, Oregon 97229-5892

*Trademarks: UNIX, AT&T Bell Laboratories, Inc.; XENIX, Microsoft, Inc.; VAX, Digital Equipment Corporation

Circle no. 275 on reader service card.

Write-Hand Man

"Almost a Sidekick for CP/M"

Ted Silveira—Computer Currents, Aug. 27, 1985

"WHM is ingenious and works as intended"
Jerry Pournelle, BYTE Magazine, Sept. 1985 (c) McGraw-Hill

Now available for CP/M 2.2, CP/M 3.0 and ZRDOS!

The convenience of *Sidekick* on your CP/M machine! Trigger **Write-Hand-Man** with a single keystroke and a window pops open to run desk accessories. Exit **Write-Hand-Man** and both the screen and program are restored. Use with any CP/M program and most any CP/M machine. Takes only 5K of memory.

FEATURES

Notepad for quick notes	File and Directory viewer
Appointment calendar	Quick access phonebook
HEX calculator	14 digit decimal calculator

BONUS

Add applications written by you or others! No other "*Sidekick*" lets you add applications. Dump screens, setup printers, communicate with other computers, display the date and time. Let your imagination run wild!

\$49.95 (California residents add tax), shipping included. COD add \$2. Sorry, no credit cards or purchase orders. 30 day guarantee. Formats: 8 inch IBM, Northstar and most 5 inch (please specify).

Write-Hand-Man only works with CP/M 2.2, ZRDOS and CP/M 3.0 (please specify). Simple terminal configuration required. Not available for TurboDOS. Compatible with keyboard extenders, hard disks, and other accessories.

Poor Person Software

3721 Starr King Circle
Palo Alto, CA 94306
415-493-3735

Trademarks: **Write-Hand-Man** — Poor Person Software, CP/M—Digital Research, *Sidekick*—Borland International

Circle no. 169 on reader service card.

Productivity Tools from SCE

• **The Seidl Make Utility (SMK)** is the most powerful make utility you can buy for MS-DOS. When changes are made to any program module, SMK will issue only those commands *necessary and sufficient* to rebuild the system. SMK uses a super fast, proprietary dependency analysis algorithm that analyzes all dependencies before rebuilding any files. SMK understands complicated dependencies involving nested include files, source, and object code libraries. A high-level dependency definition language makes setting up dependencies easy. It supports parameterized macros, local variables, for loops, constants, include files, command line parameters, in-line and block comments, full pathname and directory support, and more! Works with most compilers, assemblers, and linkers. **\$99.95**

• **The Seidl Version Manager (SVM)** is a state of the art version control system for MS-DOS. It maintains a complete revision history of any text file, without duplication, and can rebuild any previous version of the file. SVM has highly optimized compression routines included for large projects and archiving. Other features include: full pathname and directory support, wildcards, exception cases, on line help, typeset documentation, tutorial, automatic error recovery, and much more. SVM is ideal for all software and text development projects. No other version management system delivers the features, performance, and reliability of SVM. **\$299.95**

• **SMK+SVM** together form an extremely powerful, fully integrated set of productivity tools. **\$379.95**

SEIDL COMPUTER ENGINEERING

1163 E. Ogden Ave., Suite 705-171
Naperville, IL 60540
(312) 983-5477

Circle no. 114 on reader service card.

CACHE22 + CP/M 2.2 = CP/M Max!

CACHE22 is a front-end system program that buries all of CP/M 2.2 in banked memory. It helps 8080/Z80 computers to survive by providing up to 63.25K of TPA plus the ability to speed disk operations, eliminate system tracks, and run Sidekick-style software without loss of transient program space. Complete source and installation manual, \$50.00.

CP/M is a trademark of Digital Research Inc.
Sidekick is a trademark of Borland International



MIKEN OPTICAL COMPANY

53 Abbott Avenue, Morristown, NJ 07960
(201) 267-1210



Circle no. 261 on reader service card.



to **C**

the dBx translator

- **dBx** produces quality **C** direct from dBASE II or III programs.
- Move dBASE programs to UNIX or other machines.
- Improve program speed and reliability.
- Support multi-user/network applications.
- With power guidebook of conversion hints.
- Includes full screen handler and uses your current **C** database manager.
- May be used to move existing programs or help dBASE programmers learn **C** easily.
- For MSDOS, PCDOS, UNIX, XENIX, Macintosh, AMIGA. (Uses ANSI.SYS driver on MSDOS, CURSES under UNIX)
- Priced from \$350, also available from distributors.

Desktop A.I.
1720 Post Rd. E. #3
Westport, CT 06880
(203) 255-3400

Circle no. 258 on reader service card.

FTL Modula-II

\$49.95!



Your next computer language. The successor to Pascal, Modula is powerful. Why? Once a routine is written, it need never be recompiled. Programs work everywhere from Z80 through VAX.

FTL Modula-II is a full Z80 CP/M compiler (MSDOS version soon)! It's fast -- 18K source compiles in 7 seconds! The built-in split screen editor is worth \$60 alone. Some standard features: full recursion, 15 digit reals, CP/M calls, coprocessors, assembler and linker. The one-pass compiler makes true Z80.COM, ROMable code, too. Get the language you've waited for now. Only \$49.95!

FTL Editor Toolkit

Full source to our split-screen programming editor. Curious? Want to customize to your tastes? Want sample Modula-II code? This is perfect for you. Comes with all you need for your personal editor or terminal installer. Just \$39.95!

Workman and Associates
112 Marion Avenue
Pasadena, CA 91106
(818) 796-4401

We have over 200 formats in stock! Please specify your format when ordering. Add \$2.50 per order for shipping. We welcome COD orders!

Circle no. 244 on reader service card.

Time and Money.

We've just done something we know you'll like. We've made the SemiDisk far more affordable than ever before. With price cuts over 25% for most of our product line. Even our new 2 megabyte units are included.

It's Expandable

SemiDisk Systems builds fast disk emulators for more microcomputers than anyone else. S-100, IBM-PC, Epson QX-10, TRS-80 Models II, 12, and 16. You can start with as little as 512K bytes, and later upgrade to 2 megabytes per board...at your own pace, as your needs expand. Up to 8 megabytes per computer, using only four bus slots, max! Software drivers are available for CP/M 80, MS-DOS, ZDOS, TurboDOS, VALDOCS 2, and Cromix. SemiDisk turns good computers into great computers.

SEMI DISK

SemiDisk Systems, Inc., P.O. Box GG, Beaverton, Oregon 97075

	<u>512K</u>	<u>1Mbyte</u>	<u>2Mbyte</u>
SemiDisk I, S-100	\$695	\$1395	
SemiDisk II, S-100	\$995		\$1995
IBM PC, XT, AT	\$595		\$1795
QX-10	\$595		\$1795
TRS-80 II, 12, 16	\$695		\$1795
Battery Backup Unit	\$150	\$150	\$150

Someday you'll get a SemiDisk.
Until then, you'll just have to....wait.

Call 503-646-5510 for CBBS/NW, 503-775-4838 for CBBS/PCS, and 503-649-8327 for CBBS/Aloha, all SemiDisk equipped computer bulletin boards (300/1200 baud). SemiDisk, SemiSpool trademarks of SemiDisk Systems.

Circle no. 85 on reader service card.



CONCURRENCY

LISTING TWO (Listing continued, text begins on page 36)

```

enqueue2(CR);
enqueue2(LF);
end;
LF: {ignore};
CC: exit;
BS,RUB:
begin
enqueue2(BS);
enqueue2(SPACE);
enqueue2(BS);
end;
CQ: fifos[2].flow:=true;
CS: fifos[2].flow:=false;
else enqueue2(b);
end{case};
until false;{forever!}
exit;
end;
procedure initall;
var
  i: integer;
begin
  sp1:=0;
  sp2:=0;
  sp3:=0;
  oldn:=0;
end;
Procedure schedule;
begin
  if oldn=TASKS then nextn:=1
  else nextn:=oldn+1;
end;
procedure defer;
var sp: integer;
begin
  case oldn of
    0: sp0:=stackptr;
    1: sp1:=stackptr;
    2: sp2:=stackptr;

```

```

  3: sp3:=stackptr;
  end{case};
  schedule;
  oldn:=nextn;
  case nextn of
    0: sp:=sp0;
    1: sp:=sp1;
    2: sp:=sp2;
    3: sp:=sp3;
  end{case};
  if sp<>0 {initialized}
  then begin
    stackptr:=sp;
  end
  else {not initialized}
  begin
    case nextn of
      1: keyboard;
      2: filter;
      3: print;
    end{case};
  end;
end{defer};
begin{main}
initall;
writeln('<Demonstration of multitasking with queues (FIFOs)>');
writeln;
writeln('Control-S stops output (you can still type ahead!)');
writeln('Control-Q restarts output (you see what you have
typed ahead)');
writeln('RUB or BACKSPACE will "undo" on screen the last
letter');
writeln('Control-C terminates this program');
writeln;
defer;
writeln('main: done');
end.

```

End Listings

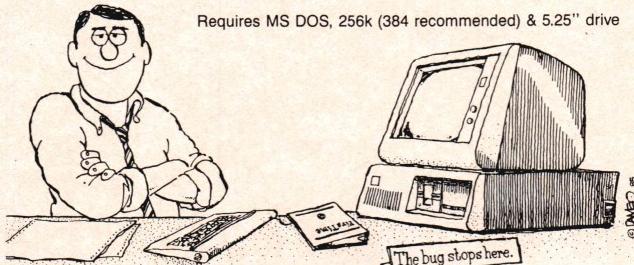
The First Idea-Processor For Programmers.

FirsTime™

Has features no other editor has.

- Fast program entry through single keystroke statement generators
- Fast editing through syntax oriented cursor movements
- Dramatically reduced debugging time through immediate syntax checking.
- The error checking is thorough and includes semantics • Undefined variables, types and constants • Assignment statements with mismatched types • Errors in include files and macro expansions
- Automatic program formatter (you specify the rules)
- Split Screen editing Command DOS from FirsTime
- Reading a file with errors moves cursor automatically to point of error
- Unique programmer-oriented features
 - zoom command gives top-down view of program logic
 - view macro command shows expansion of a C macro in the editor
 - view/update include file allows you to view and update an include file
 - transform command allows you to transform statements to related ones
 - search for next error command

Requires MS DOS, 256K (384 recommended) & 5.25" drive



To Order Call: (201) 741-8188 or write:

SPRUCE TECHNOLOGY CORPORATION



P.O. Box 7948
Shrewsbury, NJ 07701

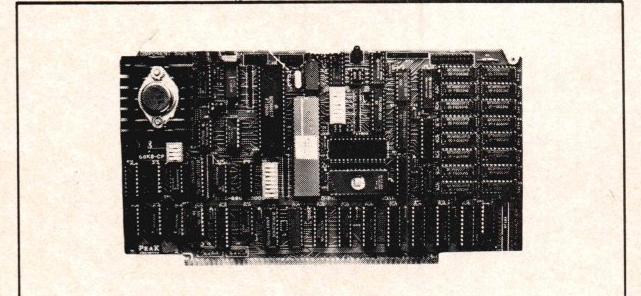
FirsTime for Turbo Pascal \$ 74.95

FirsTime for dBase III \$125.00

FirsTime for MS-Pascal \$245.00

FirsTime for C \$295.00

FirsTime is a trademark of Spruce Technology Corporation • MS-DOS is a trademark of Microsoft Corporation • IBM is a trademark of International Business Machines Inc. • Turbo Pascal is a trademark of Borland International • dBase III is a trademark of Ashton-Tate



68K8-CP

Expand Your System with a 68000 CoProcessor

Peak Electronics' 68K8-CP is a high performance 68000 software development package designed to easily integrate into your existing S-100 system. The package consists of the 68K8-CP coprocessor card, CP/M-68K, and a software toolkit that includes a UNIX V7 compatible floating point C compiler and a symbolic debugger.

Any system running CP/M®-2.2, CP/M-3.0 or CP/M-86 can be running CP/M-68K within minutes without any change in existing hardware or software. This card does not replace your current processor. All of the original system's devices (RAM, disks, and other peripherals) are immediately available to the user of CP/M-68K. All files can be accessed by whichever operating system is currently active. Control is transferred between operating systems with a simple one line command.

Features:

- Does not replace your current CPU card or software
- Includes CP/M-68K with UNIX® V7 compatible floating point C compiler and a symbolic debugger
- All developed C and Assembly code is fully relocatable and ROMable
- 8 or 10MHz CPU with no wait state RAM
- 128K bytes of RAM expandable to 512K
- 2 serial and 1 parallel I/O ports
- IEEE-696-1983, S-100 Compatible
- 30 day money back guarantee
- 1 year parts and labor warranty

Complete Package: \$995.00
VISA or Master Card Accepted



P.O. Box 700112, San Jose, CA 95170-0112
(408) 233-5108

Circle no. 164 on reader service card.

Circle no. 266 on reader service card.

SPEEDING MS DOS

LISTING ONE (Text begins on page 44)

```
;-----;
; Weissman/Dr. Dobbs submission Listing 1:   ;
; Assembler BIOS diskette read test program.  ;
;-----;

;-----;
; Assembler code to test diskette times,   ;
; reading 16 tracks through INT 13.        ;
; 10/1/85 Gregg Weissman                   ;
;          E-X-E Software Systems           ;
;-----;

HI_RES_TIMER    equ 1

; Set this to 1 if you implement
; the 1024 tick/sec AT timer.
; It will return time in 1k'ths of
; a second. If using the usual IBM
; 18.2 ticks per second timer, set to 0
; and receive counts at that rate.

if HI_RES_TIMER

  GET_TIME      equ 10h          ; Define the proper AH function value
                                ; For int 1ah

else

  GET_TIME      equ 0

endif

;-----;
; Start the test code:                  ;
;-----;

CODE segment
assume cs:code,ds:code,es:code,ss:code

org 80h

TIME_LO dw ?           ; Time accumulator out of the way.
TIME_HI dw ?

org 100h

START: ;-----;
; First, read a dummy sector to start the ;
; diskette: this way motor start time will ;
; not affect the timings.                 ;
;-----;

mov ax,0201h          ; Read, 1 sector.
mov cx,1              ; Starting @ track 0, sector 1
mov dx,1              ; Side 0, diskette B:
mov bx,offset DATA_AREA
int 13h              ; Perform the op.
jb STOP               ; Error: abort.

;-----;
; Now, get the time of day and perform   ;
; the test. Read 32 tracks, 16 cyl's.    ;
;-----;

cli
mov ah,GET_TIME
int 1ah
mov TIME_LO,dx
mov TIME_HI,cx
sti
xor bp,bp             ; Set a register to count reads.
mov di,290             ; number of sectors total in testfile.
mov dx,1              ; disk B:
mov cx,0001            ; Start at trk 0, sect 1
mov bx,offset DATA_AREA

SIDE_1:
mov al,9              ; Number not mod 9, might do partial
cmp di,al              ; read on last track.
jnb N2                ; Move partial read value into al
mov ax,di

N2:
mov ah,2              ; Read disk function code into AH
int 13h              ; Perform disk read.
jb STOP               ; Error: don't bother with re-try.

sub di,9              ; Subtract default count of sectors.
jbe DONE              ; We're done if di <= 0
inc bp                ; Count the op. to verify number done.
```

```

xor    dh,1          ; Flip sides.
jne    SIDE_1        ; If we flipped to side 1, do it.
inc    ch            ; else bump to next track (cylinder)
jmp    SIDE_1        ; and go back for more.

DONE:
    mov    ah,GET_TIME   ; OK, stop the clock.
    int    1ah
    sub    dx,TIME_LO    ; Get end_time - Start_time.
    sbb    cx,TIME_HI

STOP:
; WARNING!
; This program does not return to DOS!
; Do not run this except under DEBUG!

    int 3              ; DEBUG breaks here, read time in DX
; BP=20h and Carry Clear if no errors.

org    200h

DATA_AREA      label byte

CODE    ends
end    START

```

End Listing One

LISTING TWO

```

;-----;
; Weissman/Dr. Dobbs Submission. Listing 2: ;
; AT timer handler.                         ;
;-----;

;-----;
; Memory-resident handler for the 1024      ;
; tick/sec. realtime clock in the IBM AT:   ;
;-----;
; Enables INT 70, the clock interrupt, and    ;
; counts the ticks at the 1k rate. The range is ;
; 2**32 div 2**10, or 2**22 seconds. Long enough. ;

```

(Continued on next page)

Wizard C

*Discover the powers of Wizard C
for yourself!*

"...written by someone who has been in the business a while. This especially shows in the documentation."

Computer Language
February, 1985

"Wizard's got the highest marks for support."

"The Wizard compiler had excellent diagnostics; it would be easier writing portable code with it than with any other compiler we tested."

Dr. Dobb's Journal
August, 1985

Full Lint checking, six memory models, 8087 support, in-line assembly language, ROMable data support, full library source code. Cross-compilers are available on VAX/VMS and UNIX machines.

(617) 641-2379

Only \$450.



11 Willow Court
Arlington, MA 02174



Circle no. 116 on reader service card.

QUICK REF

Indexing at your fingertips! Take the pressure off the tedious search for that much needed article.

Quick Ref offers rapid recall of magazine articles by title, author, and key combinations.

FOR THE INTRODUCTORY PRICE OF

\$34.95

QUICK REF
PROVIDES:

- * Key access to magazine articles
- * Rapid search by title, author, and key combinations
- * Convenient on-line help
- * Easy to use manual

FREE INTRODUCTORY OFFER WITH PURCHASE

The completely indexed
Dr. Dobb's Journal
Data Base

Available for IBM Compatibles and Victor 9000

Terra Base Software
906 S. 8th Street

Laramie, Wyoming 82070

(800) 238-4790 9:00 A.M.—5:00 P.M. M.S.T.

All orders shipped U.P.S. Surface shipping included in price.
Visa/MasterCard accepted. Foreign orders please add \$15.00. Checks must be on U.S. Bank in U.S. dollars. Specify machine and DOS version.

Circle no. 231 on reader service card.

SPEEDING MS DOS

LISTING TWO (Listing continued, text begins on page 44)

```

; Sets up the usual INT 1A (Get/set time of day) ;
; to handle get/set of the hi-resolution timing ;
; data via special codes in AH register. ;
;
; With AH=10h, caller receives current time in 1k ;
; ticks. ;
; CX returns high 16 bits, DX the low. ;
; AH=11h allows user to set the 32-bit count. ;
; CX=High count to set, DX is low. ;
;
; 10/1/85 Written by Gregg Weissman ;
; E-X-E Software Systems ;
; Released to the public domain. ;
;-----;

CODE segment
assume cs:CODE,ds:CODE,es:CODE,ss:CODE

;-----;
; Conserve memory space: maintain variables ;
; in the unused areas of the program prefix. ;
;-----;

org 5ch

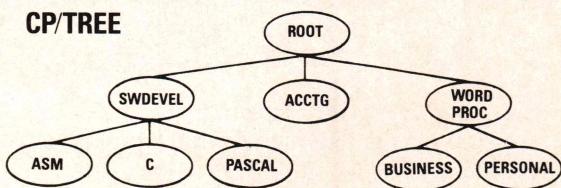
I1AJMP dd ? ; Vector to usual 1A int.
I70JMP dd ? ; Ditto the INT 70.
TIMELO dw ? ; low 16 bits of time.
TIMEHI dw ? ; Hi 16 bits of time.

org 100h ; Start of .COM program

BEGIN: jmp short START ; Jump around handlers.

```

CP/TREE



Tree-Structured Named Directories for CP/M 2.2

- Transforms user areas into Unix-like directories
- Provides Unix-like directory commands:
CD, MKDIR, PWD, RMDIR, TREE
- Includes a CCP replacement featuring:
 - Command and file search path. Use programs like WordStar from any directory!
 - Erase with query
 - Wildcard rename with query
- Provides output redirection to disk file
- Uses as little as $\frac{1}{2}$ k RAM, never more than $\frac{1}{4}$ k
- A must for hard disks
- Installs easily: requires no modifications to BDOS or BIOS
- Requires standard CP/M 2.2 (not 3.0 or Apple), Z80, 48k RAM

\$29.95 plus \$4.00 s&h

To order: Specify disk format (8" SSSD, NorthStar DD. Call for info on others.). MC, Visa, COD (add \$1.90), check (delays shipping 2 weeks). MA residents add 5% sales tax. POs not accepted.

Precise Electronics

P.O. Box 339
New Town Branch
Boston, MA 02258
tel: (617) 332-3977

AppleTM Apple Computer. CP/M[®] Digital Research. WordStar[®]
MicroPro International. Z80[®] Zilog. Unix[®] AT&T Technologies.

C CODE FOR THE PC

source code, of course

Concurrent C	\$45
Coder's Prolog in C	\$45
LEX	\$25
YACC & PREP	\$25
Small-C compiler for 8088	\$20
tiny-c interpreter & shell	\$20
Xlisp 1.5a & tiny-Prolog	\$20
C Tools	\$15

The Austin Code Works

11100 Leafwood Lane
Austin, Texas 78750-3409
(512) 258-0785

Free shipping on prepaid orders

No credit cards

Circle no. 250 on reader service card.

```

;-- Handle the Get/Set time-of-day INT 1A:    ;
; As mentioned, AH=10h Returns time,          ;
; AH=11h Sets time                         ;
; Illegal values >11h return Carry status ;
;

INT_1A proc far
assume ds:nothing, es:nothing, ss:nothing

    cmp     ah,10h           ; See if it's our range of cmnd vals.
    jb      NOPE_1A          ; If less than, assume BIOS int 1a.
    sub     ah,10h           ; Remove special code "bias" of 10h
    je      GET_TIME         ; and branch if equal to "Get" cmnd.
    dec     ah               ; See if "Set" cmnd.
    jne     BAD_CMD_1A       ; If not, branch to error.

    ;-- set time: cx,dx

    mov     TIMEHI,cx
    mov     TIMELO,dx
    iret

GET_TIME:
    mov     cx,TIMEHI        ; Obtain the number of times the 1024
    mov     dx,TIMELO         ; tick per sec. interrupt has occurred.

BAD_CMD_1A:
    stc
    ret     2                ; Return an error flag.

NOPE_1A:
    jmp     I1AJMP           ; Continue to BIOS handler via vector.

INT_1A endp

```

(Continued on next page)

We're Ready,
Are You?

**68000
68010 68020**

WE ARE PROUD TO ANNOUNCE THE BIRTH OF THE NEWEST MEMBERS
OF OUR 68000 FAMILY . . . YOUR 68020 TOOLS ARE HERE!

TOOL KIT

- 68000/10/20 Assembler Package:
 - Macro Cross/Native Assembler
 - Linker and Librarian
 - Cross Reference Facility
 - Symbol Formatter Utility
 - Object Module Translator
- Green Hills C 68000/10/20 Optimizing Compilers
- Symbolic Debuggers

FEATURES

- Written in C; fast, accurate, portable.
- Supports 68000 and 68010.
- 5,000 line test suite included.
- EXORmacs compatible.
- Produces full listings and maps.
- Outputs S-records and Tek-Hex formats.
- Runs native or cross. Extensive libraries.
- Supports OASYS compilers.
- Generates PROMable output and PIC.
- Full Floating Point support.

AVAILABILITY

VAX, microVAX, 8600, Sun, Pyramid, Masscomp, IBM/PC, OASYS Attached Processors for VAX and PC, others. Runs under VMS, Bsd 4.2, System V, MS/DOS, dozens more.

You name it . . .

We provide a "One-Stop Shopping" service for more than 100 products running on, and/or targeting to, the most popular 32-, 16- and 8-bit micros and operating systems.

A DIVISION OF XEL

Oasys

60 Aberdeen Avenue, Cambridge, MA 02138 (617) 491-4180

We Specialize In:

Cross/Native Compilers C, Pascal, Fortran, Cobol, Basic, APL, PL/I, Prolog, Lisp, ADA — Assemblers/Linkers — Symbolic Debuggers — Simulators — Interpreters — Translators — Converters — Profilers — QA Tools — Design Tools — Comm. Tools — OS Kernels — Editors — Spreadsheets — Data Bases — VAX & PC Attached Processors and more

We Support:

680xx, 80x86, 320xx, 68xx, 80xx, dozens more

Circle no. 254 on reader service card.

SPEEDING MS DOS

LISTING TWO (Listing continued, text begins on page 44)

```
;-----;
; Handle the 1024 ticks/second realtime      ;
; clock interrupt: maintain 32-bit count.   ;
;-----;
; Still assume ds, etc. nothing in effect.   ;
;-----;

INT_70 proc far

    push    ax
    push    ds
    mov     ax,cs
    mov     ds,ax

assume ds:CODE

    inc     TIMELO           ; Bump low counter.
    jne     CONT_70          ; Continue if count not wrapped around
    inc     TIMEHI           ; Bump high counter.

CONT_70:

    pushf
    call    I70JMP           ; Simulate INT nn
    call    ENABLE_INT        ; Call the realtime clock handler,
                           ; to let BIOS handle user-event-wait.
    pop    ds
    pop    ax
    iret

INT_70 endp

;-----;
; This routine enables the "PIE", or          ;
; periodic interrupt event. It is called      ;
; when the system initializes, and after      ;
; each tick, in case BIOS turned it off.      ;
;-----;
; This communicates with the CMOS            ;
; module via ports 70h & 71h. The             ;
; "JMP $+2" is for a delay between           ;
; port accesses, allowing the device         ;
; time to settle down. The 80286 is          ;
; too fast.                                    ;
;-----;

ENABLE_INT proc near

    cli
    mov    al,0bh             ; Prevent interruption.
    out    70h,al             ; First, identify the reg.
    jmp    $+2                ; we want (0b is the one).
    in     al,71h             ; Just a long NOP.
    or    al,01000000b        ; This bit is int. enable
    mov    ah,al              ; Save value: we need al
    mov    al,0bh             ; Address the register again.
    out    70h,al             ; NOP
    mov    al,ah              ; Get back the desired value.
    out    71h,al             ; Now stored in CMOS & ready.
    in     al,0ah              ; Int. control chip register.
    and    al,0feh             ; Bit 0 is another int. enable
    out    0ah,al              ; now set to allow interrupt.
    mov    al,20h              ; Send "End of interrupt" for
    out    0ah,al              ; good measure.
    out    020h,al             ; ditto controller chip 1.

    sti
    ret

ENABLE_INT endp

;-----;
; Initialize the interrupt handlers and       ;
; the interrupt generator itself. Leave       ;
; behind the resident code.                   ;
;-----;
; We leave out amenities like DOS version   ;
; test and check for an actual AT.           ;
;-----;

START:
assume ds:CODE,es:CODE,ss:CODE

    mov    ah,10h             ; First check to see if already
    clc
    mov    cx,1234h           ; resident,
    mov    dx,cx              ; by constructing an unlikely
    int    1ah                ; time value of 12341234h
    cmp    cx,dx              ; See if we get the time back.
                           ; If cx=1234=dx, not present.
```

```

je      NOT THERE
mov    dx,offset ALREADY      ; Tell the user it's in already
mov    ah,9                      ; using DOS print function.
int    21h
mov    ax,4c01h                  ; Terminate with errorlevel 1
int    21h

NOT THERE:
mov    ax,3570h                  ; Get the current vectors
int    21h
mov    word ptr I70JMP,bx        ; from DOS.
mov    word ptr I70JMP+2,es       ; Save in prefix are variables.
mov    ax,35lah                  ; Got the 70, now get 1A
int    21h
mov    word ptr I1AJMP,bx
mov    word ptr I1AJMP+2,es

push   cs                         ; Restore es for form's sake
pop    es

mov    THIS_SEG,cs

cli
mov    dx,offset INT_1a          ; Set interrupt vectors to our ha
mov    ax,251ah
int    21h
mov    dx,offset INT_70          ; Ditto for int 70.
mov    ax,2570h
int    21h
mov    dx,offset HELLO          ; Print a message
mov    ah,9
int    21h

call   ENABLE_INT                ; Start the timer.

mov    dx,offset START+15        ; Set terminate address in
mov    cl,4                      ; paragraphs, rounding up &
shr    dx,cl                      ; dividing by 16.

mov    TIMELO,0                  ; Clear the current time.
mov    TIMEHI,0

mov    ax,3100h                  ; Terminate & stay put.
int    21h                        ; End.

```

(Continued on next page)

-C Source Code

RED

Full Screen Text Editor

- RED is fast! RED uses all of your terminal's special functions for best screen response. RED handles files as large as your disk automatically and quickly.
 - RED is easy to use for writers or programmers. RED's commands are in plain English.
 - RED comes with complete source code in standard C. RED has been ported to mainframes, minis and micros.
 - RED comes with a Reference Card and a Reference Manual that provides everything you need to use RED immediately.
 - RED is unconditionally guaranteed. If for any reason you are not satisfied with RED your money will be refunded promptly.

RED: \$95
Manual: \$10



Call or write today for
more information:

Edward K. Ream
1850 Summit Avenue
Madison, WI 53705

To order:

Either the BDS C compiler or the Aztec CII compiler is required for CP/M 80 systems. Digital Research C compiler v1.1 is required for CP/M 68K systems. No compiler is required for IBM or Kaypro systems.

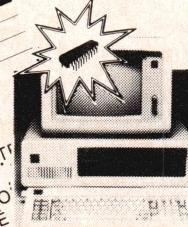
Specify both the machine desired (IBM, Kaypro or CP/M) and the disk format described (8 inch CP/M single density or exact type of 5½ inch disk).

Send a check or money order for \$95 (\$105 U.S. for foreign orders). Sorry, I do NOT accept phone, credit card, or COD orders. Please do not send purchase orders unless a check is included. Your order will be mailed to you within one week.

Dealer inquiries invited

Circle **no. 90** on reader service card.

RUN/CPM™



ONLY \$99.95
INCLUDING HARDWARE!
Plus over 20
CP/M80 programs!

PUT A CP/M COMPUTER IN YOUR PC!

And run 1,000's of CP/M programs
up to 30% faster, directly from your CP/M disks!

How does it work?
RUN/CPM virtually transforms your PC into any of the most popular CP/M systems. A simple registration converts your PC's 8088/86 microprocessor with our CPM-2030 microprocessor gives your computer the ability to run both 8 bit CP/M and 16 bit MS-DOS programs. RUN/CPM will transform your PC's floppy drives into CP/M drives able to directly read, write, and format over 100 CP/M disks. Terminal emulation supporting terminals of all the popular terminals completes the transformation of your PC into a CP/M system.

Performance?
Depending on the application, many of your CP/M programs will run up to 30% faster on your PC. Other programs made available to run CP/M programs in color, logical and physical drive assignments, run CP/M or MS-DOS programs from the same prompt. RUN/CPM is the solution to running CP/M software on PCs.

1-800-637-7226
Orders only.

Micro Interfaces Corporation
6824 N.W. 169th Street, Miami, Florida 33105
(305) 823-8088
Telex 5106004680 MICRO INTER CO
Ask About Our Intel Operating System Interfaces

OLEM, VAR, Dealers, Inquiries Invited

Circle no. 110 on reader service card.

SPEEDING MS DOS

LISTING TWO (Listing continued, text begins on page 44)

```
;-- Messages for the user:  
HELLO db      'Hi-resolution AT timer now loaded.',13,10,'$'  
ALREADY db    'The resident timer handler is already installed.',13,10,'$'  
CODE      ends  
end      BEGIN
```

End Listing Two

LISTING THREE

```
;-----  
; Weissman/Dr. Dobbs Submission LISTING 3:  
; Sample program fragment to change diskette:  
; BIOS parameters.  
;  
;  
; Define the structure of the BIOS diskette parms;  
;  
DISKPARMS  struc  
SPEC1      db      ?          ; Don't change these values  
SPEC2      db      ?  
SPEC3      db      ?  
SPEC4      db      ?  
SPEC5      db      ?  
SPEC6      db      ?  
SPEC7      db      ?  
SPEC8      db      ?  
SPEC9      db      ?  
HEAD_SETTLE db      ?  
MOTOR_WAIT  db      ?  
DISKPARMS  ends  
;  
;  
; Define the location of the pointer to the parms;  
;  
SYS0      segment at 0000  
org      78h  
DISK_PTR   label dword  
SYS0      ends
```

```
;-----  
; Here is fragment to access the current  
; parameter block, and set the settle-time  
; and motor wait.  
;  
; Enter with the settle time in AL, and  
; the motor-wait value (in 1/8ths of a sec.)  
; in AH.  
;  
; So you can preserve and restore the current  
; values, AL & AH return with them: you must  
; save them for later if you want to restore.  
;  
SET_PARMS  proc  
push bx           ; Save the reg's used.  
push ds  
xor bx,bx  
mov ds,bx  
assume ds:SY0  
lds bx,DISK_PTR  
assume ds:nothing  
xchg [bx].HEAD_SETTLE,al  
xchg [bx].MOTOR_WAIT,ah  
pop ds  
pop bx  
ret  
SET_PARMS  endp  
;  
;-- You need your own "assume" and "end" statements!
```

End Listing Three

PC/VI

Full Screen Editor for MS-DOS (PC-DOS)

Looking for an Ultra-Powerful Full-Screen editor for your MS-DOS or PC-DOS system? Are you looking for an editor FULLY COMPATIBLE with the UNIX* VI editor. Are you looking for an editor which not only runs on IBM-PC's and compatibles, but ANY MS-DOS system? Are you looking for an editor which provides power and flexibility for both programming and text editing? If you are, then look no further because **PC/VI IS HERE!**

The following is only a hint of the power behind **PC/VI**: English-like syntax is command mode, mnemonic control sequences in visual mode; full undo capability; deletions, changes and cursor positioning on character, word, line, sentence, paragraph or global basis; editing of files larger than available memory; powerful pattern matching capability for searches and substitutions; location marking; joining multiple lines; auto-indentation; word abbreviations and MUCH, MUCH MORE!

The **PC/VI** editor is available for IBM-PC's and generic MS-DOS based systems for only \$149. For more information call or write:

Custom Software Systems
P.O. Box 551 MO
Shrewsbury, MA 01545
617-842-1712

The UNIX community has been using the VI editor for years. Now you can run an implementation of the same editor under MS-DOS. Don't miss out on the power of **PC/VI!**

*UNIX is a trademark of AT&T Bell Laboratories.

Circle no. 268 on reader service card.

QPARSER™

Translator Writing System

THE PRODUCTIVITY TOOL FOR SOFTWARE DEVELOPERS

QPARSER assists you in writing:

- * Compilers * Translators * Interpreters *
- * Prototypes * Simulators * Syntax Checkers *
- * Data Converters * Assemblers *

QPARSER is a unique LALR(1) parser generator:

Generates complete source code for your application in C, Pascal, or another language of your choice; Extensive examples include a Pascal subset compiler, assembler, and simulator;

The widely used college text, *Compiler Construction: Theory & Practice*, from SRA Associates, was written by the author of QPARSER

Lauded by both industrial and university users Available for: IBM PC,XT,AT; DEC VAX; HP 9816; MACINTOSH (PC System \$400; Demo \$10; Educational/Site Licenses available)

"LEADERS IN SOFTWARE TOOLS"



1164 Hyde Ave., San Jose CA 95129

Toll-free Orders: (800) 538-9787; In CA: (408) 727-6671

Circle no. 264 on reader service card.

LISTING FOUR

Timing the DOS command: "COPY B:TESTFILE.001 NUL:"
DOS200LS LOGDOS/GETSTATS (1.1) DOS timing statistics output.
Function #: Count: Total Time: Mean Time:
3D 3 0:2.1970 0:0.7323 (OPEN)
3F 3 0:7.4149 0:2.4716 (READ)

Other DOS function timings not related to the test are left out

STATISTICS SUMMARY:

Total DOS function calls counted:

83

Total DOS execution time:

0:9.7218

Mean DOS execution time:

0:0.1171

Histogram of the relevant time taken up by different functions - as before, irrelevant DOS calls left out.

3D: 22.6%*****

3F: 76.3%*****

End Listing Four

LISTING FIVE

BASIC A Interpreter with default 128-byte buffer size.

DOS200LS LOGDOS/GETSTATS (1.1) DOS timing statistics output.
Function #: Count: Total Time: Mean Time:
3D 1 0:2.1970 0:2.1970 (OPEN)
3F 1153 0:41.963 0:0.0364 (READ)

STATISTICS SUMMARY:

Total DOS function calls counted:

1171

Total DOS execution time:

0:44.215

Mean DOS execution time:

0:0.0378

(Continued on next page)

Now! Automatic time and date stamping for CP/M® 2.2

DATESTAMPER™

- Extends CP/M 2.2 to automatically record date and time a file is created, read and modified. DateStamper reads the exact time from a real-time clock if you have one, or records the order in which you use files each day.
- Datestamping information is contained in a separate file and read by our directory utility, SDD.COM. Powerful DATSWEEP file management utility also included.
- Simple menu-driven installation. Large library of clocks supplied, with provision to add others. Many options configurable to your individual requirements.
- Disks prepared for datestamping are fully compatible with standard CP/M. DateStamper also runs with all versions of ZCPR, Z-RDOS and many other CP/M-80 modifications.

CP/M is a registered trademark of Digital Research, Inc.

Avoid erasing the wrong files!

Back up files by time and date!

"DateStamper... is a real winner."

Bruce Morgen, Users Guide, Jul-Aug. 1985

Write or call for further information
(714) 659-4432

Plu'Perfect Systems

8" SSSD, Kaypro, Osborne, H/Z-89 formats	\$49
(Most other formats, add \$5)	
Shipping & handling	\$3
California residents add 6% sales tax	
MasterCard & Visa accepted	

BOX 1494, IDYLLWILD, CA 92349

Now available For the computer experimenter!

COMPUTER CONNOISSEUR'S DELIGHT!

NOW BE IN CONTROL WITH YOUR COMPUTER — THE ONLY PUBLICATION OF ITS KIND WRITTEN FOR THE USER. DISCOVER THE SECRETS AND LEARN THE VERSATILITY OF MODERN COMPUTER COMMAND CONTROL CONCEPTS. EXPERIMENT WITH COMPUTER AND TELEPHONE SYSTEMS, INTERFACE THEM, LEARN HOW THEY WORK, WHAT THEY DO... AND HOW TO GET THEM TO WORK FOR YOU! A COMPLETE TELEPHONE ENGINEERING COURSE IS INCLUDED IN MONTHLY CHAPTERS, BRINGING YOU THROUGH STEP, CROSSBAR, ESS, BUBBLE, AND ATOMIC SWITCHING SYSTEMS! EXCLUSIVE COVERAGE IN BIOLOGICAL COMPUTING SYSTEMS, TOO! COMPUTERS AND TELEPHONES ARE THE FUTURE. THIS PUBLICATION IS AN ABSOLUTE MUST FOR EVERYONE INTERESTED.

UNPUBLISHED MATERIAL

WIT COMICS DIRECTORY LISTING NET-WORKS ACCESS CODES

The one you've all been waiting for

NOW AVAILABLE — Learn how to repair telephones and telephone systems, how they work, in monthly installments with the magazine for you.

Computel™

PUBLISHED MONTHLY

ONE YEAR SUBSCRIPTION \$14.00
(SAMPLE COPY \$2.00)
SUBSCRIPTION & 2 PROGRAMS \$20.00

COMPUTEL—the complete SOURCE for everyone. You can now do the things you've only heard about, right in the privacy of your own home. Indispensable reference to phreaks and hackers. Learn how to get all kinds of computer programs FREE. Get the inside story of big business systems—their quirks and flaws—and remain up to date with vital occurrences within the computer industry. Computel is a publication designed for everyone who has an intense curiosity of computer systems, containing a wealth of hard to find information, codes, and numbers. Published monthly.

Computel Publishing Society
6354 VAN NUYS BL., #161-A / VAN NUYS, CA 91401

Circle no. 193 on reader service card.

Circle no. 225 on reader service card.

SPEEDING MS DOS

LISTING FIVE (Listing continued, text begins on page 44)

3D: 05.0%**
3F: 94.9%*****

End Listing Five

LISTING SIX

ASSEMBLER code calling DOS with record size of 200h bytes
DOS2TOOLS LOGDOS/GETSTATS (1.1) DOS timing statistics output.
Function #: Count: Total Time: Mean Time:
3D 1 0:1.9773 0:1.9773
3F 289 1:0.8025 0:0.2104

STATISTICS SUMMARY:

Total DOS function calls counted:

293

Total DOS execution time:

1:2.7798

Mean DOS execution time:

0:0.2143

3D: 03.1%**

3F: 96.9%*****

End Listing Six

Disclose Service Won't Make You Nervous

Technical support, personal service, competitive prices.

Disclose full service quality tested diskette duplication, packaging, documentation production and processing ensures precise duplication, thorough quality control, and expedient response to your requirements.

NOclone state of the art hardware based copy protection is true piracy protection for authorized allotments only. Each application is uniquely encrypted. Install routines are coded for nontransferrable hard disk allotments.

Commitment dates are guaranteed. Fast turnover

- up to 1000 in 24 hours, any format.
- up to 10,000 in one week, any format.

disclose
q2clone

DISCLOSE SOFTWARE PRODUCTION SERVICES

1585 North Fourth Street, San Jose, California 95112
(408) 947-1161 OUTSIDE CA: 1-800-826-4296

Circle no. 204 on reader service card.

d/MULTI MULTIUSER dBASE for TurboDOS

TRUE File and Record Locking as easy as d-BASE-II. Unlimited users can perform the magic of dBASE in the program or interactive mode

- * TurboDOS 1.3 or 1.4
- * No Peeks or Pokes
- * System Date and Time Functions
- * Printspooler Controls up to 16 printers

Martian Technologies . . .

.CREATEing Multi-users from
Single-users around the world

CALL FOR DETAILS

Martian Technologies
8348 Center Dr., Ste.-F, La Mesa, CA 92041
(619) 464-2924



Circle no. 189 on reader service card.



Continuing the Tradition DR. DOBB'S JOURNAL BOUND VOLUMES

Vol. 1 1976

The material brought together in this volume chronicles the development in 1976 of Tiny BASIC as an alternative to the "finger blistering," front-panel, machine-language programming. This is always pertinent for the bit crunching and byte saving, language design theory, home-brew computer construction and the technical history of personal computing. Topics include: Tiny BASIC, the (very) first word on CP/M, Speech Synthesis, Floating Point Routines, Timer Routines, Building an IMSAI.

Vol. 2 1977

These issues offer refinements of Tiny BASIC, plus then state-of-the-art utilities, the advent of PILOT for microcomputers and a great deal of material centering around the Intel 6080, including a complete operating system. Products just becoming available for reviews were the H-8, KIM-1, MITS BASIC, Poly Basic, and NBL. Articles are about Lawrence Livermore Lab's BASIC, Alpha Micro, String Handling, Cyphers, High Speed Interaction, I/O, Tiny Pilot & Turtle Graphics, many utilities.

Vol. 3 1978

This volume brings together the issues which began dealing with the 6502, with mass-market machines and languages to match. The authors began speaking more in terms of technique, rather than of specific implementations; because of this, they were able to continue laying the groundwork industry would follow. These articles relate very closely to what is generally available today. Languages covered in depth were SAM76, Pilot, Pascal, and Lisp, in addition to RAM Testers, S-100 Bus Standard Proposal, Disassemblers, Editors.

Vol. 4 1979

This volume heralds a wider interest in telecommunications, in algorithms, and in faster, more powerful utilities and languages, innovation is still present in every page, and more attention is paid to the best ways to use the processors which have proven longevity—primarily the 8080/Z80, 6502, and 6800. The subject matter is invaluable both as a learning tool and as a frequent source of reference. Main subjects include: Programming Problems/Solutions, Pascal, Information Network Proposal, Floating Point Arithmetic, 8-bit to 16-bit Conversion, Pseudo-random Sequences, and Interfacing a Micro to a Mainframe.

Complete your reference library. Buy the entire set of Dr. Dobb's Journals from 1976 through 1983. Bound Volumes 1-8, for \$195.00. That's \$34.00 off the combined individual prices—a savings of almost 15%!

Each book in this series contains a year's worth of Dr. Dobb's Journal monthly issues, reprinted in full and combined in one comprehensive volume. The Bound Volumes will fill the gaps in your Dr. Dobb's collection and complete your reference library.

YES! Please send me the following Volumes of **Dr. Dobb's Journal**.

Payment must accompany your order.

Please charge my: Visa MasterCard American Express

I enclose Check/money order

Card # _____ Expiration Date _____

Signature _____

Name _____ Address _____
(please, no P.O. Boxes)

City _____ State _____ Zip _____

Mail to Dr. Dobb's Journal, 501 Galveston Dr., Redwood City, CA 94063

Allow 3-6 weeks for delivery.

Vol. 5 1980

Systems software reached a new level with the advent of CP/M, chronicled herein by Gary Kildall and others (DDJ's all-CP/M issue sold out within weeks of publication). Software portability became a subject of greater import, and DDJ published Ron Cain's immediately famous Small-C compiler—reprinted here in full.

Contents include: The Evolution of CP/M, a CP/M-Flavored C Interpreter, Ron Cain's C Compiler for the 8080, Further with Tiny BASIC, a Syntax-Oriented Compiler, Writing Language, CP/M to UCSD Pascal File Conversion, Run-time Library for the Small-C Compiler.

Vol. 6 1981

1981 saw our first all-FORTH issue (now sold out), along with continuing coverage of CP/M, small-C, telecommunications, and new languages. Dave Cortesi opened "Dr. Dobb's Clinic" in 1981, beginning one of the magazine's most popular features. Highlights: Information on PCNET, the Conference Tree, and The Electric Phone Book, writing your own compiler, a systems programming language, and Tiny BASIC for the 6809.

Vol. 7 1982

In '82 we introduced several significant pieces of software, including the RED text editor and the Runic extensible compiler. Two new columns, The CP/M Exchange and The 16-Bit Software Toolbox, were launched, and we devoted special issues to FORTH and telecommunications. Resident Intern Dave Cortesi delivered his famous review of JRT Pascal and wrote the first serious technical comparison of CP/M-86 and MSdos. This was also the year we began looking forward to today's generation of microprocessors and operating systems, publishing software for the Motorola 68000 and the Zilog Z8000 as well as Unix code. And in December, we looked beyond in the provocative essay, "Fifth-generation Computers."

Vol. 8 1983

DDJ truns pro. Some of the most powerful, professional programmer's tools ever published in a magazine are in this volume. The second half of Jim Hendrix's Small C Compiler (continued from Vol. 7), Ed Ream's RED screen editor, A microcomputer subset of the Defense Department's official programming language, Ada, C and Forth and 68000 software. Because the magazine increased in size in 1983, this volume is bigger and better than ever.

Vol. 1	x	\$26.75	=
Vol. 2	x	\$27.75	=
Vol. 3	x	\$27.75	=
Vol. 4	x	\$27.75	=
Vol. 5	x	\$27.75	=
Vol. 6	x	\$27.75	=
Vol. 7	x	\$30.75	=
Vol. 8	x	\$31.75	=
All 8	x	\$195.00	=
		Sub-total	\$ _____

California residents add applicable sales tax %

Postage & Handling Must be Included with order.
Please add \$2.25 per book in U.S. (\$5.25 each surface mail outside U.S. Foreign Airmail rates available on request.)

TOTAL \$ _____

SPEEDING MS DOS

LISTING SEVEN (Listing continued, text begins on page 44.)

```
ASSEMBLER code, block read size of 0FE00h
DOS2TOOLS LOGDOS/GETSTATS (1.1)      DOS timing statistics output.

Function #:   Count:          Total Time:        Mean Time:
              3D             1                0:2.1970           0:2.1970
              3F             3                0:7.1403           0:2.3801

STATISTICS SUMMARY:
Total DOS function calls counted:    7
Total DOS execution time:          0:9.3923
Mean DOS execution time:          0:1.3418

3D: 23.4%*****
3F: 76.0%*****
```

End Listing Seven

LISTING EIGHT

```
ASSEMBLER code, block read size of 2400h ( 9 k-bytes )
DOS2TOOLS LOGDOS/GETSTATS (1.1)      DOS timing statistics output.

Function #:   Count:          Total Time:        Mean Time:
              3D             1                0:2.0322           0:2.0322
              3F            17               0:9.5021           0:0.5589

STATISTICS SUMMARY:
Total DOS function calls counted:    22
Total DOS execution time:          0:11.534
Mean DOS execution time:          0:0.5243

3D: 17.6%#####
3F: 82.4%#####
```

End Listings

You Read Dr. Dobb's Journal of Software Tools and You Don't Subscribe?!

Save over \$13.00 off newsstand prices for 2 yrs.

Save over \$5.00 for 1 yr.

Can you afford to miss an issue with information vital to your interests? As a subscriber you can look forward to articles on Small-C, FORTH, MS DOS, S-100, Compiler optimization, Concurrent Programming and more, delivered right to your door. And you'll never miss the issue that covers your project.

Yes!

Sign me up for

2 yrs. \$56.97

1 yr. \$29.97

Check/money order enclosed

Charge my Visa, MasterCard,
American Express

Please bill me later

Name _____
Address _____

Credit Card _____ Zip _____

Credit Card _____

Exp. date _____

Account No. _____

Signature _____

Please allow 6-8 weeks for delivery.

MAIL TO: DR. DOBB'S JOURNAL, PO Box 27809, San Diego, CA 92128

3051

Dr. Dobb's Toolbook for C Is Here!

Dr. Dobb's Toolbook for C \$29.95

An anthology of some of the finest C articles and listings from past issues of Dr. Dobb's Journal. The Dr. Dobb's Toolbook of C includes the **Small C Compiler, macro assembler, library, preprocessor, Small-Tools, and Small-C tools for text processing.** Standard C programs include Allen Holub's grep and getargs (a general purpose command line processor) and a collection of useful subroutines from Anthony Skjellum's C/Unix Programmer's Notebook.

Order Now!

YES! Send me

Dr. Dobb's Toolbook for C \$29.95 _____
MS/PC DOS C Tools Package \$82.95 _____
CP/M C Tools Package \$99.95 _____
Tax (CA only) _____
Postage _____

(Add \$1.75 for The Toolbook,
or \$8.75 for each C Package)

TOTAL _____

For CP/M Package, specify format:

Kaypro Apple Zenith Z-100 DS/DD
 Osborne 8" SS/SD

20% Off Dr. Dobb's Special C Packages!

Special CP/M C Tools Package ONLY \$99.95

The CP/M C Package includes Dr. Dobb's Toolbook for C, Hendrix's Small C Handbook—a companion to the compiler containing its source listings; Small C Compiler on disk, source and object code; Small Tools Text Processor on disk, source code only, along with documentation in The Small Tools Manual; Small Mac Assembler on disk, source and object code, and The Small Mac Manual.

Special MS/PC DOS C Tools Package ONLY \$82.95

The MS/PC DOS C Package includes Dr. Dobb's Toolbook for C, Hendrix's Small C Handbook and MS/PC DOS Addendum—an MS/PC DOS specific companion to the compiler; Small C Compiler on disk, source and object code; Small Tools Text Processor on disk, source code only, along with documentation in The Small Tools Manual.

Check Enclosed Charge my _____ VISA M/C Amer. Exp.

Card# _____ Exp. Date _____

Name _____

Address _____

City _____ State _____ Zip _____

Offer good in U.S. only. Inquire about foreign shipping rates

C68 & C68/020 C COMPILERS for MC680X0

NOW AVAILABLE
ON IBM PC

- Produce highly optimized code
- Complete development environment: Assembler, Linking and Downline Loaders, Runtime Libraries
- Available for Motorola, DEC, and Alcyon host computers
- The #1 choice for compact, fast MC680X0 code
- \$1495 for C68 (Motorola host)
\$2295 for C68/020 (Motorola host)



5010 Shoreham Place
San Diego, CA 92122
(619) 587-1155 TELEX 5106004947

DEC is a Trademark of Digital Equipment Corporation.

Circle no. 277 on reader service card.

Excellence

In your job, it depends on having the best tools available at your disposal. With such tools, your productivity increases and your work becomes easier.

Wisely, you keep a sharp eye open for products using the latest technology...Those truly representing the state of the art.

You have now located the source of advanced debugging technology for PC-DOS and CP/M-80. More powerful debugging software is not available anywhere...at any price. Yet the cost is affordable to even the smallest budget.

DSD-80...*Absolutely the most powerful and easiest to use debugger for CP/M-80. Full screen symbolic design now includes a back tracing capability. Only 125.00*

DSD-86...*New and innovative design combines the most sophisticated user interface with the most flexible display to create a new generation of debugging technology for the IBM PC. Only 69.95*



SoftAdvances

P.O. Box 49473 - Austin, Texas 78765 - (512) 478-4763

Visa & Mastercard Accepted. Please include 4.00 for shipping and handling.

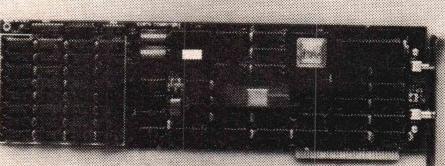
Circle no. 83 on reader service card.

LOOKING FOR AT
PERFORMANCE
FROM YOUR PC?



EARTH HAS IT FOR
LESS THAN \$1,000!

YOUR SEARCH IS OVER!! EARTH COMPUTERS' exciting new high-speed, 80286 accelerator card, **TurboACCEL-286™**, is just what you've been looking for. The **TurboACCEL-286** will boost your PC performance up to Five times...its completely software transparent...and its only \$995! **TurboACCEL-286** will function with most operating systems and application programs (unlike other so-called accelerator boards).



The **TurboACCEL-286** features a high-speed, 8MHz, 80286 processor, 512Kbytes of RAM (expandable to 1Mbytes), a switch for 8088 operation, and facilities for an 80287 math coprocessor. It occupies one expansion slot, is completely compatible with most PCs and is software transparent. End your search for AT performance. Order the **TurboACCEL-286** today! Call or write:



EARTH COMPUTERS

P.O. Box 8067, Fountain Valley, CA 92728
TELEX: 910 997 6120 EARTH FV

(714) 964-5784

Ask about EARTH COMPUTERS' other fine PC and S-100 compatible products.

Circle no. 179 on reader service card.

ASSEMBLE

LISTING ONE (Text begins on page 122)

```
/* a square root algorithm
   04/21/85, R. A. Campbell
*/
#define NUMBER 60000

long number, ndivs, nshfts;
long sqrto, sqrtt;

main() /* a test loop for square root calculation */
{
    nshfts = ndivs = 0;
    sqrto = 100;
    for (number = 0; number <= NUMBER; ++number)
    {
        sqrtt = sqrt (number);
        /* for timing, comment out next 2 lines */
        if (sqrto != sqrtt)
            printf ("sqr %6d=%4d ", number, sqrtt );
        sqrto = sqrtt;
    }
    printf ("\nTot Divs= %u Ave Divs.(x10)= %u",
           ndivs, (10 * ndivs)/NUMBER);
    printf ("\nTot Shfts= %u Ave Shfts.(x10)= %u",
           nshfts, (10 * nshfts)/NUMBER);
}

sqrt (numb) /* a 'rational' square root */
register long numb;
{ register long guess1, guess2, error;
/* return; for timing checks */
if (numb != 0)
{ guess1 = 1;
    guess2 = numb;
    while ((guess1 << 1) < guess2)
    {
        guess1 <= 1; /* multiply guess1 by 2 */
        guess2 >= 1; /* divide guess2 by 2 */
        +nshfts; /* for statistics */
    }
    do
        /* now, check */
    { guess1 += guess2; /* figure sum */
        guess1 >= 1; /* fig mean, divide by two */
        guess2 = numb / guess1; /* figure quotient */
        error = guess1 - guess2; /* figure error */
        ++ndivs; /* for statistics */
    } while (error > 0);
    return guess1;
}
else
    return 0;
}

```

End Listing One

LISTING TWO

```
;SCN:v5 (C) 1985
PROGRAM
STATIC 1
Qnumber: DOUBLE
Qndivs: DOUBLE
Qnshfts: DOUBLE
Qsqrto: DOUBLE
Qsqrto: DOUBLE
PROGRAM
JUMP Q_shell
Qmain: PROC
VAR
BEGIN
MOVQ.D 0,R6
MOV.D R6,Qndivs
MOV.D R6,Qnshfts
MOV.D 100,Qsqrto
MOVQ.D 0,Qnumber
Q1: CMP.D 60000,Qnumber
BLT.S Q3
Q2: MOV.D Qnumber,TOS
BSR Qsqrto
MOV.D R6,Qsqrto
CMP.D Qsqrto,Qsqrto
BEQ.S Q5
MOV.D Qsqrto,TOS
MOV.D Qnumber,TOS
ADDR Z6,TOS
JSR Qprintf
ADJSP -8
Q5: MOV.D Qsqrto,Qsqrto
ADDQ.D 1,Qnumber
BR.S Q1
Q3: MOV.D 10,R6
MUL.D Qndivs,R6
QUO.D 60000,R6
MOV.D R6,TOS
MOV.D Qndivs,TOS
ADDR Z6+13,TOS
JSR Qprintf
ADJSP -8
MOV.D Qnshfts,R6
QUO.D 60000,R6

```

```
MOV.D R6,TOS
MOV.D Qnshfts,TOS
ADDR Z6+4,TOS
JSR Qprintf
ADJSP -8
ENDPROC
26: BYTE 115,113,114,32,37,53,100,61,37,51,100,32
BYTE 0,10,84,111,116,32,68,105,118,115,61,37
BYTE 51,117,32,65,118,101,32,68,105,118,115,46
BYTE 40,120,49,48,41,61,37,50,117,0,10,84
BYTE 111,116,32,83,104,102,116,115,61,37,51,117
BYTE 32,65,118,101,32,83,104,102,116,115,46,61
BYTE 37,50,117,0

```

End Listing Two

LISTING THREE

```
Qsqrt: PROC ;square root, dividing version
Znumb: DOUBLE ;registers R0, R6 & R7 used by compiler
VAR ;registers R1 - R5 for register variables
BEGIN <R4,R3,R2,R1,>;push used registers
; BR Q15 ;for timing
MOV.D Znumb,R1 ;get number into R1
CMPQ.D 0,R1 ;is number zero?
BEQ.Q8 ;mustn't divide by zero
MOVQ.D 1,R2 ;guess1 = 1
MOV.D R1,R3 ;guess2 = number
Q9: MOVQ.D R2,R6 ;load guess1
ASH.D 1,R6 ;guess1 << 1
CMPQ.D R3,R6 ;compare with guess2
BLE.S Q11 ;while ( (guess1 << 1) < guess2 )
ASH.D 1,R2 ;guess1 <= 1
ASH.D -1,R3 ;guess2 >= 1
BR.S Q9 ;see if done
Q11: ADD.D R3,R2 ;guess1 += guess2
ASH.D -1,R2 ;guess1 /= 2
MOVQ.D R1,R6 ;numb
QUO.D R2,R6 ;numb / guess1
MOVQ.D R6,R3 ;guess2 = numb / guess1
MOVQ.D R2,R6 ;error = guess1 - guess2
SUB.D R3,R6
MOVQ.D R6,R4 ;error = guess1 - guess2
CMPQ.D 0,R4
BGE.S Q14 ;while ( error > 0 )
BR.S Q13
Q14: MOVQ.D R2,R6 ;return guess1
Q13: MOVQ.D 0,R6 ;return 0
Q8: ENDPROC ;exit
IMPORT Q_shell 00
IMPORT Qprintf 01
END;SCN, 03/22/85

```

End Listing Three

LISTING FOUR

```
Qsqrt: PROC ;square root, bit-shifting
version
Znumb: DOUBLE
VAR
BEGIN <R4,R3,R1,>;push registers used
; BR.S SQRT2 ;for timing
MOVQ.D Znumb,R0 ;number to R0
MOVQ.D 0,R1 ;estimate = 0
MOVQ.D 16,R4 ;loop count
SQRTL: ROT.D 2,R0 ;2 msb's to 2 lsb's of number
MOV.B R0,R3 ;get shifted number in R3
AND.B 3,R3 ;isolate 2 lsb's of R3
ASH.D 2,R1 ;shift estimate
OR.B R3,R1 ;OR 2 lsb's into estimate
ASH.D 1,R6 ;shift trial root
MOVQ.D R6,R3 ;get trial root
ASH.D 1,R3 ;trial root * 2
CMP.D R1,R3 ;compare to estimate
BLS.S SQRT2 ;need a bit?
ADDQ.D 1,R6 ;add to trial root
ADDQ.D 1,R3 ;add to estimate
SUB.D R3,R1 ;adjust estimate
SQRT2: ACB.S -1,R4,SQRTL ;count down, loop
SQRT2: ENDPROC ;return root, in R6

```

End Listings

Work Smart with These Powerful C Utilities

Get more value from your C system. Boost program quality and slash development time with these professional utilities for leading C-compiler systems.

C Utility Library \$185 \$155 Over 300 C subroutines

C and assembler source code and demonstration programs for screen handling, color printing, graphics, DOS disk and file functions, memory management and peripherals control.

C-tree \$395 \$329

B-Tree database system

Store, update and retrieve records easily. High-level multi-key ISAM routines and low-level B-Tree functions. Available for MS-DOS, CP/M-86, and CP/M-80. Easily transported. Adaptable for network and multiuser. Includes source.

PHACT \$295 \$200 Data Base Record Manager

Includes high-level features found in larger database systems. Available for MS-DOS, CP/M-86 and CP/M-80.

Pre-C \$395 \$329

LINT-like source code analyzer

Locates structural and usage errors. Cross-checks multiple files for bad parameter declarations and other interface errors.

Windows for C \$195 \$165

Versatile window utility

Supports IBM PC compatible and some non-compatible environments.

PANEL \$295 \$235

Screen generating utility

Create custom screens via simple, powerful editing commands. Select colors, sizes and types, edit fields. Includes direct input utility.

HALO \$280 \$199

Ultimate C graphics

A comprehensive package of graphics subroutines for C. Supports multiple graphics cards.

PLINK-86 \$395 \$315

Overlay linker

Includes linkage editor, overlay management, a library manager and memory mapping. Works with Microsoft and Intel object format.

To order or for information call:



(In NJ call 201-530-6307)
Circle no. 109 on reader service card.

Circle no. 109 on reader service card.

Learn and Use AI Technology In Your First Evening With PROLOG-86

A complete *Prolog Interpreter, Tutorial, and set of Sample Programs:*

Modify and write Expert Systems.

Use the simple "Guess the animal" example on the Tutorial or use the sophisticated system for Section 318 of the US Tax Code written by one of the PROLOG-86 authors and published in the March, 1985 issue of Dr. Dobb's Journal.

Understand Natural Language

Use the sample program that produces a dBase DISPLAY command as output.

Programming experience is not required, but a logical mind is. Serious development of experimental systems is practical with PROLOG-86. 1 or 2 pages in Prolog is often equivalent to 10 or 15 in C.

RECENT IMPROVEMENTS: MSDOS commands, on-line help, load Editor.

AVAILABILITY: All MSDOS, PCDOS systems.

ONLY

\$125

Full refund if not satisfied during first 30 days.

**Solution
Systems**

335-D Washington St.
Norwell, Mass. 02061
617-659-1571
800-821-2492

Circle no. 155 on reader service card.

LEARN LISP Interactively and Write "Realistic" Programs with TransLISP for Only \$75

A "COMMON LISP" compatible Tutorial, Interpreter, Debugging, and Pretty Printer plus a Fast, Full Screen Editor, Samples and Help

Start Easily and Quickly:

A complete, modular tutorial helps you learn LISP at your own pace. An integrated, interactive environment provides all of the elements needed to enter, modify, analyze and debug programs.

Natural Language, Expert Systems and Mailing List:

Natural Language concepts are illustrated by a phone number retrieval program. Choose the best word processing program for you with the Expert System. File handling and typical data processing work are demonstrated by a Mailing List program.

Write Realistic Programs:

Short examples and substantial programs of about 10 pages in length help you learn by modifying, studying and using the key concepts needed to write programs of 1000 lines or more.

The "COMMON LISP" Standard:

TransLISP includes a 230+ function subset of the "COMMON LISP" Standard. Use extras like the MSDOS interface and graphics. Or use "strict compatibility" to make programs written in TransLISP, with no changes, work with other COMMON LISP systems like VAX LISP, GC/LISP or LISP Machine LISP.

Use and Modify the Mailing List program to learn how to handle "normal" programming in LISP.

Runs on any MSDOS or PCDOS Systems: Not copy-protected, TransLISP is available in just about any 3", 5" or 8" format. PC compatibles can run TransLISP with no installation procedure. 192K memory and 1 floppy drive are the minimums required.

ONLY

\$75

Full refund if not satisfied during first 30 days.

**Solution
Systems**

335-D Washington St.
Norwell, Mass. 02061
617-659-1571
800-821-2492

Circle no. 148 on reader service card.

8080 SIMULATOR

LISTING TWO (Continued from February)

sui	move.b (pseudopc)+,d0 sub.b d0,rega move sr,d0 and.w regconff,d0 move.b 0(flagptr,d0.w),regf jmp (return)	; D6 Sui nn	subq.l #2,pseudosp lea.l \$18(targbase),pseudopc jmp (return)
rst10	move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp lea.l \$10(targbase),pseudopc jmp (return)	; D7 Rst 10	rpo btst #2,regf bne mloop move.b 1(pseudosp),d0 rol.w #8,d0 move.b (pseudosp),d0 addq.l #2,pseudosp lea.l 0(targbase,d0.1),pseudopc jmp (return)
rc	btst #0,regf beq mloop move.b 1(pseudosp),d0 rol.w #8,d0 move.b (pseudosp),d0 addq.l #2,pseudosp lea.l 0(targbase,d0.1),pseudopc jmp (return)	; D8 Rc	poph move.b (pseudosp)+,regl(regs) move.b (pseudosp)+,regh(regs) jmp (return)
nopD9	bra illegl	; D9 Illegal for 8080	jpo move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudosp btst #2,regf bne mloop lea.l 0(targbase,d0.1),pseudopc jmp (return)
jc	move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudosp btst #0,regf beq mloop lea.l 0(targbase,d0.1),pseudopc jmp (return)	; DA Jc addr	xthl move.b regl(regs),d0 move.b (pseudosp),regl(regs) move.b d0,(pseudosp) move.b regh(regs),d0 move.b 1(pseudosp),regh(regs) move.b d0,1(pseudosp) jmp (return)
in	moveq #0,d0 move.b (pseudopc)+,d0 move.l #\$ff0000,a0 move.b 0(a0,d0.1),rega jmp (return)	; DB In nn	cpo move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudosp btst #2,regf bne mloop move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp lea.l 0(targbase,d0.1),pseudopc jmp (return)
cc	move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudosp btst #0,regf beq mloop move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp lea.l 0(targbase,d0.1),pseudopc jmp (return)	; DC Cc addr	pushh move.b regh(regs),-(pseudosp) move.b regl(regs),-(pseudosp) jmp (return)
nopDD	bra illegl	; DD Illegal for 8080	ani and.b (pseudopc)+,rega move.b rega,d0 and.w regconff,d0 move.b 16(flagptr,d0.w),regf jmp (return)
sbi	asr.b #1,regf move.b (pseudopc)+,d0 moveq #0,d1 subx.b d0,rega move sr,d0 and.w regconff,d0 move.b 0(flagptr,d0.w),regf jmp (return)	; DE Sbi nn	rst20 move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp lea.l \$20(targbase),pseudopc jmp (return)
rst18	move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp)	; DF Rst 18	rpe btst #2,regf beq mloop move.b 1(pseudosp),d0 rol.w #8,d0

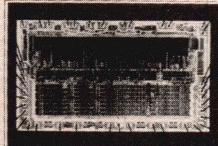
move.b (pseudosp),d0 addq.l #2,pseudosp lea.l 0(targbase,d0.1),pseudopc jmp (return)		subq.l #2,pseudosp lea.l 0(targbase,d0.1),pseudopc jmp (return)
pchl move.w regh(regs),d0 lea.l 0(targbase,d0.1),pseudopc jmp (return)	; E9 Pchl	*preED bra illegl ; ED Illegal for 8080
jpe move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudopc btst #2,regf beq mloop lea.l 0(targbase,d0.1),pseudopc jmp (return)	; EA Jpe addr	* ED is a prefix for the popular Z-80 instructions. Some support for them is provided by the minimal Z-80 simulation routines in the next file.
xchg move.w regd(regs),d0 move.w regh(regs),regd(regs) move.w d0,regh(regs) jmp (return)	; EB Xchg	xri move.b (pseudopc)+,d0 eor.b d0,rega move.b rega,d0 and.w regconf,d0 move.b 16(flagptr,d0.w),regf jmp (return)
cpe move.b 1(pseudopc),d0 rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudopc btst #2,regf beq mloop move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp)	; EC Cpe addr	rst28 move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp lea.l \$28(targbase),pseudopc jmp (return)
		rp btst #7,regf bne mloop move.b 1(pseudosp),d0 rol.w #8,d0 move.b (pseudosp),d0 addq.l #2,pseudosp

(Continued on next page)

FROM THE DEVELOPERS OF THE 65816 Microprocessor

The programming handbook you've been waiting for...

Programming the
65816 Microprocessor
Including 6502 and 65C02



0-89303-769-3/288 p/\$22.95

- Outlines the programming strengths of the 65816, 6502, and 65C02.
- Includes a review of basic concepts, architecture, and logical operations.

Now available at your local bookstore, or call toll free 1(800)624-0023 to order your copy today. In New Jersey, call 1(800)624-0024.

Brady

Circle no. 219 on reader service card.

ATTENTION

C-PROGRAMMERS

File System Utility Libraries

All products are written entirely in K&R C. Source code included, No Royalties, Powerful & Portable.

BTree Library

75.00

- High speed random and sequential access.
- Multiple keys per data file with up to 16 million records per file.
- Duplicate keys, variable length data records.

ISAM Driver

40.00

- Greatly speeds application development.
- Combines ease of use of database manager with flexibility of programming language.
- Supports multi key files and dynamic index definition.
- Very easy to use.

Make

59.00

- Patterned after the UNIX utility.
- Works for programs written in every language.
- Full macros, File name expansion and built in rules.

Full Documentation and Example Programs Included.

ALL THREE PRODUCTS FOR —

149.00

For more information call or write:

softfocus

Credit cards accepted.

1343 Stanbury Drive
Oakville, Ontario, Canada
L6L 2J5
(416) 825-0903

Dealer inquiries invited.

Circle no. 259 on reader service card.

8080 SIMULATOR

LISTING TWO (Continued from February)

<pre> lea.l 0(targbase,d0.1),pseudopc jmp (return) popp move.b (pseudosp)+,regf ; F1 Pop P move.b (pseudosp)+,rega jmp (return) jp move.b 1(pseudopc),d0 ; F2 Jp addr rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudosp btst #7,regf bne mloop lea.l 0(targbase,d0.1),pseudopc jmp (return) di jmp (return) ; F3 Di cp move.b 1(pseudopc),d0 ; F4 Cp addr rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudosp btst #7,regf bne mloop move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp lea.l 0(targbase,d0.1),pseudopc jmp (return) pushp move.b rega,-(pseudosp) move.b regf,-(pseudosp) jmp (return) oria or.b (pseudopc)+,rega ; F6 Ori nn move.b rega,d0 and.w regconff,d0 move.b 16(flagptr,d0.w),regf jmp (return) rst30 move.l pseudopc,d1 ; F7 Rst 30 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp lea.l \$30(targbase),pseudopc jmp (return) rm btst #7,regf ; F8 Rm beq mloop move.b 1(pseudosp),d0 rol.w #8,d0 </pre>	<pre> move.b (pseudosp),d0 addq.l #2,pseudosp lea.l 0(targbase,d0.1),pseudopc jmp (return) sphl move.w regh(regs),d0 ; F9 Sphl lea.l 0(targbase,d0.1),pseudosp jmp (return) jm move.b 1(pseudopc),d0 ; FA Jm addr rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudosp btst #7,regf beq mloop lea.l 0(targbase,d0.1),pseudopc jmp (return) ei jmp (return) ; FB Ei cm move.b 1(pseudopc),d0 ; FC Cm addr rol.w #8,d0 move.b (pseudopc),d0 addq.l #2,pseudosp btst #7,regf beq mloop move.l pseudopc,d1 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp lea.l 0(targbase,d0.1),pseudopc jmp (return) nopFD bra illegl ; FD Illegal for 8080 cpi cmp.b (pseudopc)+,rega ; FE Cpi nn move sr,d0 and.w regconff,d0 move.b 0(flagptr,d0.w),regf jmp (return) rst38 move.l pseudopc,d1 ; FF Rst 38 sub.l targbase,d1 move.b d1,-2(pseudosp) rol.w #8,d1 move.b d1,-1(pseudosp) subq.l #2,pseudosp lea.l \$38(targbase),pseudopc jmp (return) .end </pre>
---	---

End Listing Two

LISTING THREE

```

*****
*           *
* This file contains the special Z-80 simulation routines and      *
* the Morrow HDC/DMA support routines.                                *
*           *
*****
```

```

globl preED,outspec
xdef mloop,illegal

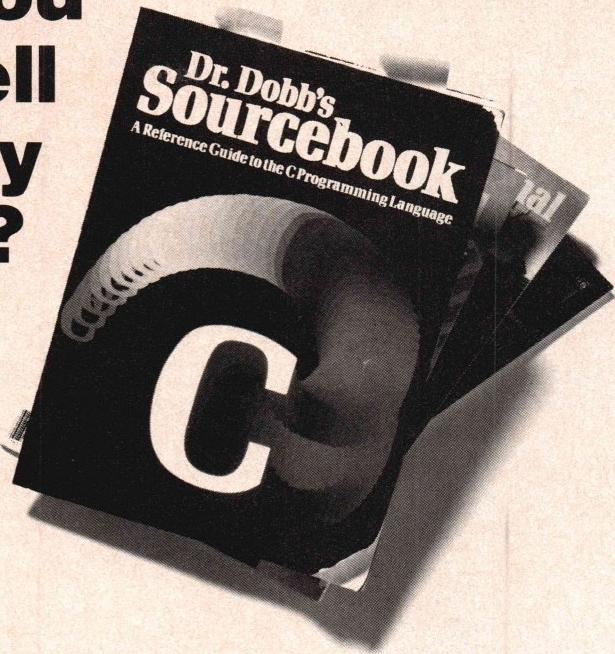
return equ @16,r      ; JMP (return) is fast return to MLOOP.
pseudopc equ @15,r    ; 8080's PC is register A5.
opptr   equ @14,r    ; Pointer to opcode dispatch table.

```

(Continued on page 112)

Who Says You Can't Tell A Book By Its Cover?

***Dr. Dobb's Sourcebook:
A Reference Guide
for the C Programming
Language***



For years, serious programmers have relied on Dr. Dobb's Journal for the technical tools of their trade. Now, Dr. Dobb's presents the definitive programmers guide to the who, what, where, when and why of C, the leading language among software developers. This comprehensive guide to new information, products and services specific to C will be your most often-used reference!

In this valuable guide you'll find:

- An extensive directory of hardware and software services—including classes and seminars, C programming opportunities, and on-line services
- A bibliography with over 300 listings of available articles and books on C
- A comprehensive C product listing—including C compilers, graphics modules, utilities, editors and development systems, and more!
- And much more practical C programming information

At only \$7.95, no C programmer can afford to be without this unique reference.

To order by credit card, call toll free: 1-800-528-6050 ext. 4001. Ask for item 004 or mail this coupon along with payment to: **Dr. Dobb's Journal, 501 Galveston Dr. Redwood City.**

PAYMENT MUST ACCOMPANY YOUR ORDER

I enclose check/money order

Please charge my VISA M/C American Express

Card # Exp. Date

Signature

Name

Address

City State Zip

Please send me copies of **Dr. Dobb's Sourcebook**

at \$7.95 each =

+ Shipping & Handling =

(Must be included with order. Please add \$1.50 per book in U.S. \$3.25 each surface mail outside U.S. Foreign airmail rates available on request.)

TOTAL =

3113E

YOU NEED A GOOD C LIBRARY



COMPLETE SOURCES NO ROYALTIES

COMPREHENSIVE C Power Packs include over 1000 functions which provide an integrated environment for developing your applications efficiently. "This is a beautifully documented, incredibly comprehensive set of C Function Libraries."

— Dr. Dobb's Journal, July 1984

USEFUL "...can be used as an excellent learning tool for beginning C Programmers..."

— PC User's Group of Colorado, Jan. 1985

FLEXIBLE Most Compilers and all Memory Models supported.

RECOMMENDED "I have no hesitation in recommending it to any programmer interested in producing more applications code, using more of the PC capabilities, in much less time." — Microsystems, Oct. 1984

■ PACK 1: Building Blocks I \$149
DOS, Keyboard, File, Printer, Video, Async

■ PACK 2: Database \$399
B-Tree, Virtual Memory, Lists, Variable Records

■ PACK 3: Communications \$149
Smartmodem™, Xon/Xoff, X-Modem, Modem-7

■ PACK 4: Building Blocks II \$149
Dates, Textwindows, Menus, Data Compression, Graphics

■ PACK 5: Mathematics I \$99
Log, Trig, Random, Std Deviation

■ PACK 6: Utilities I \$99
(EXE files)
Arc, Diff, Replace, Scan, Wipe
Master Card/Visa, \$7 Shipping, Mass. Sales Tax 5%

ASK FOR FREE DEMO DISKETTE

NOVUM
ORGANUM
INC.

SOFTWARE
HORIZONS
INC.

44 Mall Rd., Burlington, MA 01803
(617) 273-4711

Circle no. 262 on reader service card.

8080 SIMULATOR

LISTING THREE (Continued from February)

```

pseudosp equ 013,r ; 8080's SP is register A3.
flagptr equ 012,r ; Pointer to 8080's flag lookup table is A2.
targbase equ 011,r ; Pointer to 8080's address space is A1.
regs equ 011,r ; Base pointer to 8080's registers is A1.

regcon0e equ 7,r ; Register based constant #SE (for speed).
regcon01 equ 6,r ; Register based constant #$1.
regcon0f equ 5,r ; Register based constant #$F.
regcon0d equ 4,r ; Register based constant #$FF.
regf equ 3,r ; 8080's Flags
rega equ 2,r ; 8080's Accumulator

regb equ -8 ; Offsets from register base pointer for
regc equ -7 ; 8080's pseudo-registers.
regd equ -6 ; A & F are in Data Registers.
rege equ -5 ; Pseudo-PC is kept in an Address Register.
regh equ -4
regl equ -3
regop1 equ -2 ; Operand 1 for DAA storage
regop2 equ -1 ; " 2 " "

```

```

data
page
*****
*          *
*      Opcode dispatch table. One longword entry per opcode of the   *
*      target (Z-80) processor, including illegals.                   *
*          *
*****          *
*          *      Only a few of the most popular instructions are simulated.   *
*          *      Support for the Z-80 Block move instructions is provided   *
*          *      as the flags for this simulation resemble those of the Z-80   *
*          *      rather than the 8080. Certain packages (notably BDS-C) check   *
*          *      the flags and mistakenly assume a Z-80, then use LDIR/LDDR.   *
*          *      Therefore, minimal Z-80 support is provided for these   *
*          *      instructions. By no means is this a complete simulation   *
*          *      of the Z-80.                                         *
*          *
*****          *

```

```

even
EDoptab dc.l 0,0,0,0,0,0,0,0 ; ED00
dc.l 0,0,0,0,0,0,0,0
dc.l 0,0,0,0,0,0,0,0 ; ED10
dc.l 0,0,0,0,0,0,0,0
dc.l 0,0,0,0,0,0,0,0 ; ED20
dc.l 0,0,0,0,0,0,0,0
dc.l 0,0,0,0,0,0,0,0 ; ED30
dc.l 0,0,0,0,0,0,0,0
dc.l 0,0,0,0,0,0,0,0 ; ED40
dc.l 0,0,0,0,0,0,0,0
dc.l 0,0,0,0,0,0,0,0 ; ED50
dc.l 0,0,0,0,0,0,0,0
dc.l 0,0,0,0,0,0,0,0 ; ED60
dc.l 0,0,0,0,0,0,0,0
dc.l 0,0,0,0,0,0,0,0 ; ED70
dc.l 0,0,0,0,0,0,0,0
dc.l 0,0,0,0,0,0,0,0 ; ED80
dc.l 0,0,0,0,0,0,0,0
dc.l 0,0,0,0,0,0,0,0 ; ED90
dc.l 0,0,0,0,0,0,0,0
dc.l 0,0,0,0,0,0,0,0 ; EDA0
dc.l 0,0,0,0,0,0,0,0
dc.l 1dir,cpir,0,0,0,0,0,0 ; EDB0
dc.l lddr,0,0,0,0,0,0,0 ; EDC0
dc.l 0,0,0,0,0,0,0,0 ; EDD0
dc.l 0,0,0,0,0,0,0,0 ; EDE0
dc.l 0,0,0,0,0,0,0,0

```

(Continued on page 114)

Fatten Your Mac for \$5.00

Thanks to Macintosh owners everywhere, *Dr. Dobb's* January 1985 issue #99 was a runaway best-seller.

Now, due to popular demand, the Doctor has reprinted the sought-after **Fatten Your Mac** article from the sold-out January issue. The article explains how you can pack a full 512K of memory into your system, and save half the cost by performing the upgrade yourself.

To order: Enclose \$5.00 for each copy with this coupon and send to:

Dr. Dobb's Journal, 501 Galveston Dr.
Redwood City, CA 94063

Outside U.S., add \$2.00 per copy for shipping & handling.

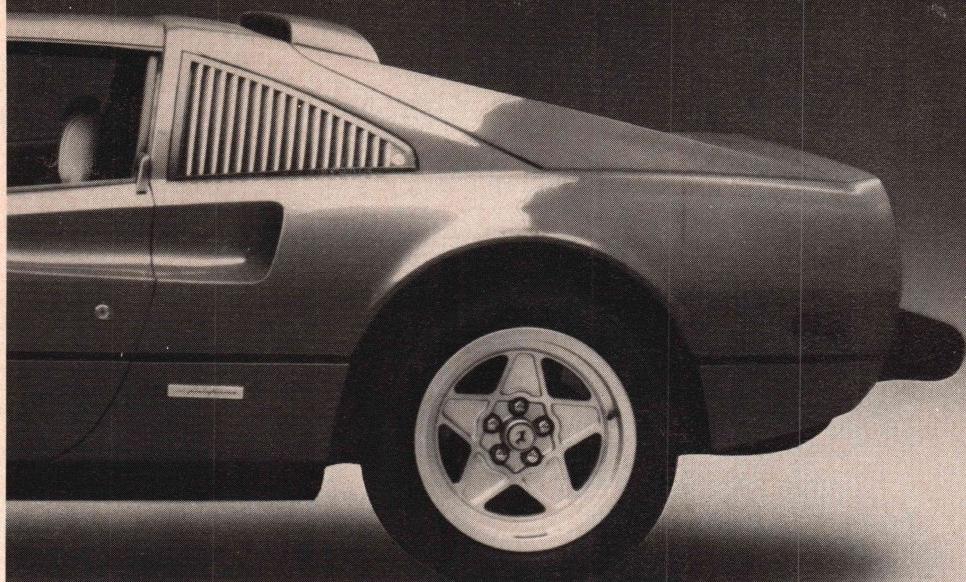
Please send me _____ copies of **Fatten Your Mac. ALL REPRINT ORDERS MUST BE PREPAID.**

Name _____

Address _____

City _____ State _____ Zip _____

3113G



New Advanced Symbolic Debugger Handles Complex Overlaid Programs.

Phoenix's new Pfix™86 Plus for MS-DOS®/PC DOS environments features configurable menus and keys, tracebacks, EGA support, and the ability to stop, save, and restore a debugging session at any time.

You can debug without a listing, since you can see and enter symbolic names or absolute addresses in breakpoints, data displays, expressions, or with the in-line assembler.

Display source files, disassembled object, data area, stack, breakpoint settings, CPU and coprocessor registers, simultaneously. Set breakpoints in the source file window. Synchronize the source display with the disassembled object. Add symbols incrementally during the debug session. And write the disassembly to disk.

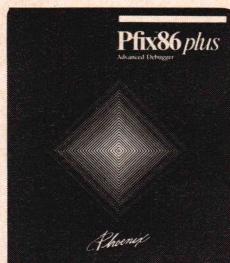
Pfix 86 Plus supports 80xx and 80xxx CPUs and floating point processors, EGA boards, user-assignable string and numeric variables, up to 100-step tracebacks, and logs to disk or printer.

\$395 complete.

Send for your Pfix 86 Plus information kit today. Call or write:

Phoenix Computer Products Corporation
320 Norwood Park South
Norwood, MA 02062
(800) 344-7200. In Massachusetts (617) 762-5030

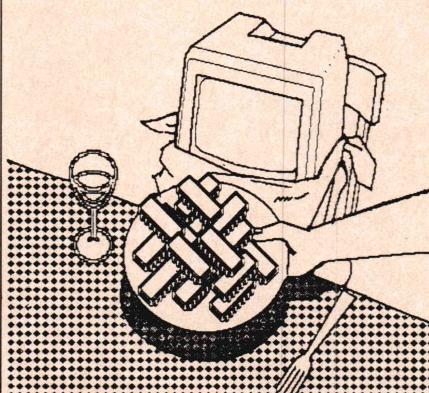
Programmers' Pfantases™ by



Phoenix

Circle no. 139 on reader service card.

Pfix86 Plus is a trademark of Phoenix Software Associates, Ltd.
MS-DOS is a registered trademark of Microsoft Corporation.



C Users' Group

Over 75 volumes of public domain software including:

- compilers
- editors
- text formatters
- communications packages
- many UNIX-like tools

Write or call for more details

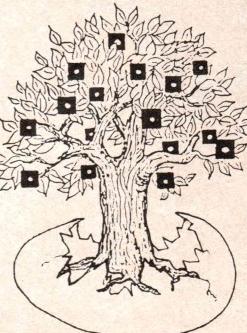
The C Users' Group

Post Office Box 97
McPherson, KS 67460
(316) 241-1065

Circle no. 181 on reader service card.

Tree Shell

A Graphic Visual Shell for Unix/Xenix End-Users and Experts Alike!



Dealer inquiries welcomed.

COGITATE

"A Higher Form of Software"
24000 Telegraph Road
Southfield, MI 48034
(313) 352-2345
TELEX: 386581 COGITATE USA

Circle no. 81 on reader service card.

Dr. Dobb's Journal
Subscription Problems?
No Problem!



Give us a call and we'll straighten it out. Today.

Outside California
CALL TOLL FREE: 800-321-3333
Inside California
CALL: 619-485-6535 or 6536

8080 SIMULATOR

LISTING THREE (Continued from February)

```

dc.l 0,0,0,0,0,0,0,0 ; EDF0
dc.l 0,0,0,0,0,0,0,0

page
text
preED moveq #0,d1 ; Zero-fill high bits.
move.b (pseudopc)+,d1 ; Grab next opcode.
asl #2,d1
lea.l EDoptab,a0
move.l 0(a0,d1.w),-(sp) ; Do the operation.
beq illgED
rts

illgED move.l (sp)+,d1 ; Trash the address,
dec.l pseudopc ; fix PPC for ILLEGAL.
bra illegl

page
*****
*          *
*      Z-80 opcode simulation routines. *
*          *
*****
```

```

ldir    move.l d2,-(sp) ; Grab count,
        move.w regb(regs),d0 ; adjust for DBRA later.
        dec.w d0
        moveq #0,d1
        moveq #0,d2
        move.w regh(regs),d1 ; Grab source.
        move.w regd(regs),d2 ; Grab dest.
        move.l a5,-(sp) ; Need an address reg.
        lea.l 0(targbase,d2.1),a5
        lea.l 0(targbase,d1.1),a0
ldirlop move.b (a0)+(a5)+ inc.w d1
           inc.w d2
           dbra d0,ldirlop
           move.l (sp)+,a5
           move.w d1,regh(regs)
           move.w d2,regd(regs)
           move.w #0,regb(regs)
           moveq #0,regf
           move.l (sp)+,d2
           jmp (return)

laddr   move.l d2,-(sp) ; Grab count,
        move.w regb(regs),d0 ; adjust for DBRA later.
        dec.w d0
        moveq #0,d1
        moveq #0,d2
        move.w regh(regs),d1 ; Grab source.
        move.w regd(regs),d2 ; Grab dest.
        move.l a5,-(sp) ; Need an address reg.
        lea.l 1(targbase,d2.1),a5
        lea.l 1(targbase,d1.1),a0
laddrlop move.b -(a0),-(a5) dec.w d1
           dec.w d2
           dbra d0,laddrlop
           move.l (sp)+,a5
           move.w d1,regh(regs)
           move.w d2,regd(regs)
           move.w #0,regb(regs)
           moveq #0,regf
           move.l (sp)+,d2
           jmp (return)

cpir    page
        move.w regb(regs),d0 ; Grab count,
        dec.w d0 ; adjust for DBRA later.
        moveq #0,d1
```

```

move.w regh(regs),d1           ; Grab source.
lea.l 0(targbase,d1.l),a0
cpirlp inc.w d1
cmp.b (a0)+,rega
dbeq d0,cpirlp
seq regf
move.w d1,regh(regs)          ; Restore result registers.
inc.w d0
move.w d0,regb(regs)
tst.b regf
bne cpirl
moveq #0,regf
jmp (return)
tst d0
beq cpirl2
moveq #$44,regf
jmp (return)
cpirl2 moveq #$40,regf
jmp (return)

page
*****
*          Output instruction simulator for Morrow HDDMA
*
*****
```

outspec

```

move.l d3,-(sp)
cmp.b #$55,d0
beq hdstart
move.l #hdbuf,d1           ; Start command? Do it,
move.l #$50,a0             ; else build first link to host buffer
move.b d1,(a0)+             ; if it's a HDRESET command.
ror.l #8,d1
move.b d1,(a0)+
ror.l #8,d1
move.b d1,(a0)+

move.l #$ff0000,a0           ; Do the output to HDDMA.
adda.l d0,a0
move.b rega,(a0)
move.l (sp)+,d3
jmp (return)
```

hdstart

```

move.l dmalink,a0
adda.l targbase,a0
moveq #0,d1
move.b (a0)+,d1
ror.l #8,d1
move.b (a0)+,d1
ror.l #8,d1
move.b (a0)+,d1
rol.l #8,d1
rol.l #8,d1
move.l d1,a0
adda.l targbase,a0
move.l a6,-(sp)
lea.l hdbuf,a6
move.w #3,d1
```

hdloop

```

move.b (a0)+,(a6)+           ; Move target buffer to host buffer, do
dbra d1,hdloop               ; all appropriate patching of addresses.
moveq #0,d3
move.b (a0)+,d3
ror.l #8,d3
move.b (a0)+,d3
ror.l #8,d3
move.b (a0)+,d3
rol.l #8,d3
rol.l #8,d3
add.l targbase,d3
move.b d3,(a6)+
ror.l #8,d3
move.b d3,(a6)+
ror.l #8,d3
move.b d3,(a6)+
move.w #5,d1
```

hdloop2

```

move.b (a0)+,(a6)+           ; Fix DMA address to -> target area.
dbra d1,hdloop2
```

; Move rest of command buffer.

MetaScope: The Debugger

MetaScope gives you everything you've always wanted in a debugger:

- **Multiple Windows**
Open and close, move through memory, display data or disassembled code.
- **Full Symbolic Capability**
Read symbols from files, define new ones, use anywhere.
- **Powerful Expression Evaluation**
Use any standard assembler operators or number formats.
- **Direct to Memory Assembler**
Enter instruction statements for direct conversion to code in memory.
- **and More!**
Log file for operations and displays, breakpoint and trace execution, modify/search/fill memory, etc.

MetaScribe: The Editor

MetaScribe has the features you need in a program editor:

- **Full Mouse Support**
Use for text selection, command menus, scrolling — or use key equivalents when more convenient.
- **Multiple Undo**
Undo all commands, one at a time, to level limited only by available memory.
- **Sophisticated Search/Replace**
Regular expressions, forward/backward, full file or marked block.
- **Multiple Windows**
Work with different files or different portions of the same file at one time.
- **Keystroke Macros**
Record keystroke sequences or predefine, assign to keys you choose.
- **and More!**
Copy between files, block copy/move/delete, set tabs and margins, etc.

MetaTools I

A comprehensive set of tools to aid your programming (full source included):

MetaMake	Program maintenance utility.
Grep	Sophisticated pattern matching utility.
Diff	Source file compare.
Filter	Text file filter.
Comp	Simple file compare.
Dump	File dump utility.
MetaSend	Amiga to PC file transfer.
MetaRecv	PC to Amiga file transfer.

Metadigm products are designed to fully utilize the capabilities of the Amiga™ in helping you develop your programs. If you're programming the Amiga, you can't afford to be without them.

MetaScope	\$95.00
MetaScribe	\$85.00
MetaTools	\$69.95

(California residents +6%).
Visa/MasterCharge accepted.

Amiga is a trademark of Commodore-Amiga Inc.

Metadigm, Inc.

19762 MacArthur Blvd., Suite 300
Irvine, CA 92715
(714) 955-2555

Circle no. 220 on reader service card.

ANNOUNCING! DR. DOBB'S COMPLETE TOOLBOX OF



New, from M&T Publishing and
Brady Communications . . .

Dr. Dobb's Toolbook for C

Item #005

A comprehensive library of valuable C code.

Many of Dr. Dobb's most popular articles on C from sold-out issues are updated and reprinted in this unique reference, along with new C programming tools, compilers, line editors, assemblers, text processing programs, and more! Dr. Dobb's Journal offers The Toolbook in a special hardbound edition for only \$29.95. You'll find:

- J.E. Hendrix's famous Small C Compiler v. 2 and A New Library for Small C (also available on disk)
- Never before published: Hendrix's new Small-Mac: An Assembler for Small C and Small Tools: Programs for Text Processing (both available on disk)
- Ron Cain's original A Small-C Compiler for the 8080's
- Plus many useful programming tools in C

Also from M&T Publishing and
Brady Communication—
**The Small-C
Handbook**
Item #006 or #006A

The Small-C Handbook by James Hendrix is a valuable resource of information about the Small-C Compiler. In addition to descriptions of the language and the compiler, The Handbook also contains the entire source listings of the compiler and its routines. A perfect companion to the Hendrix Small-C compiler offered by Dr. Dobb's on disk, The Handbook even tells how to use the compiler to generate new versions of itself. All this is in The Handbook for only \$17.95, Item #006; only \$22.95 for The Handbook with the MS/PC DOS Handbook Addendum, Item #006A.

While both The Toolbook and The Handbook provide documentation for the Small-C Compiler on disk, The Handbook provides more complete documentation and is available with an Addendum for MS/PC DOS versions.

DR. DOBB'S C TOOLS ON DISK

To complement The Toolbook, Dr. Dobbs' also offers the following programs on disk. Source code is included, and except where indicated, both CP/M and MS/PC-DOS versions are available.

Small C Compiler

Item #007

Jim Hendrix's Small-C Compiler is the most popular piece of software published in Dr. Dobbs' 10-year history. Like a home study course in compiler design, the Small-C Compiler and The Small-C Handbook provide all you need to learn how compilers are constructed, as well as teaching the C language at its most fundamental level. The Small-C Compiler is available for \$19.95 in both CP/M and MS/PC DOS versions.

Both The Toolbook and The Small-C Handbook provide documentation for the compiler; however, The Handbook provides more complete documentation for both versions, and the MS/PC DOS Small-C Handbook Addendum is recommended in addition to The Handbook for MS or PC DOS specific documentation.

Special Packages—20% Off!

Now, for almost 20% off the individual product prices, you can order a complete set of Dr. Dobbs' C programming tools for your CP/M or MS/PC DOS system!

CP/M C Package Only \$99.95 Item #005A

Ordered individually, these C tools sell over \$120! Order the CP/M C Package now and you'll receive Dr. Dobbs' Toolbook, The Small-C Handbook, The Small-C compiler on disk, The Small Tools TextProcessor on disk, along with documentation in The Small Tools Manual, The Small-Mac Assembler on disk, and The Small-Mac Manual—all for only \$99.95!

MS/PC DOS C Package Only \$82.95 Item #005B

These tools sell for over \$100! Order now and you'll receive Dr. Dobbs' Toolbook, The Small-C Handbook with the MS/PC DOS Addendum, The Small-C Compiler on disk, The Small Tools TextProcessor on disk, along with documentation in The Small Tools Manual—all for only \$82.95!

Small-Mac: An Assembler for Small-C

Item #012A

Small-Mac is a macro assembler designed to stress simplicity, portability, adaptability, and educational value. The package features simplified macro facility, C-language expression operators, descriptive error messages, object file visibility, and an externally defined machine instruction table. Included programs are: macro assembler, linker editor, load-and-go loader, library manager, CPU configuration utility, and dump relocation files. This program is available with documentation in the Small Mac Manual for \$29.95. For CP/M systems only.

Small Tools: Programs for Text Processing

Item #010A

This package consists of programs designed to perform specific functions on text files, including: editing, formatting, sorting, merging, listing, printing, searching, changing, translating, copying and concatenating, encrypting and decrypting, replacing spaces with tabs and tabs with spaces, counting characters, words, or lines, and selecting printer fonts. Source code only is included. This program is available with documentation in the Small Tools Manual for \$29.95. Both CP/M and MS/PC DOS versions available.

Books	Dr. Dobbs' Toolbook	\$29.95
	Small-C Handbook	\$17.95
	Small C Handbook with MS/PC DOS Addendum	\$22.95
Disks	Check Format	
	CP/M	\$19.95
	MS/PC DOS	
		\$29.95
		\$29.95
		\$99.95
		\$82.95
SPECIAL HOLIDAY PACKAGES	CP/M C Package	Tax
	MS/PC DOS C Package	Shipping
		TOTAL
		Subtotal
		%

CA residents add applicable sales tax
Add \$1.75 per item for shipping in U.S., \$4.25
outside U.S. Add \$8.75 shipping in U.S.
for special C Packages.

For CP/M system disks only, please specify one of the following formats:
 Kaypro Apple Zenith Z-100 DS/DD Osborne 8" SS/SD
 Inquire about other formats.

PAYMENT MUST ACCOMPANY YOUR ORDER.

Check Enclosed
 Charge my VISA M/C Amer. Exp.

Card # _____

Signature _____

Name _____

Address
(Please use street address)

City _____

State _____

Zip _____

Please return this coupon along with your payment to: Dr. Dobbs' Journal, 501 Galveston Dr. Redwood City, CA 94063. For credit card orders, call toll-free: 1-800-528-6050 Ext. 4001. Refer to item numbers 3113A

PRIME FEATURES

- Execute DOS level commands in HS/FORTH, or execute DOS and BIOS functions directly.
- Execute other programs under HS/FORTH supervision.
(editors debuggers file managers etc)
- Use our editor or your own.
- Save environment any time as .COM or .EXE file.
- Eliminate headers, reclaim space without recompiling.
- Trace and decompile.
- Deferred definition, execution vectors, case, interrupt handlers.

HS
FORTH

- Full 8087 high level support.
Full range transcendental functions (tan sin cos arctan logs exponentials)
- Data type conversion and I/O parse/format to 18 digits plus exponent.
- Complete Assembler for 8088, 80186, and 8087.
- String functions - (LEFT RIGHT MID LOC COMP XCHG JOIN)
- Graphics & Music
- Includes Forth-79 and Forth-83
- File and/or Screen interfaces
- Segment Management
- Full megabyte - programs or data
- Fully Optimized & Tested for:
IBM-PC XT AT and JR
COMPAQ and TANDY 1000 & 2000
(Runs on all true MSDOS compatibles!)
- Compare
BYTE Sieve Benchmark jan 83
HS/FORTH 47 sec BASIC 2000 sec
with AUTO-OPT 9 sec Assembler 5 sec
other Forths (mostly 64k) 55-140 sec
FASTEAST FORTH SYSTEM
AVAILABLE.
TWICE AS FAST AS OTHER
FULL MEGABYTE FORTHS!
(TEN TIMES FASTER WHEN USING AUTO-OPT!)
HS/FORTH, complete system only: \$270.

 Visa  Mastercard

**HARVARD
SOFTWARES**

P.O. BOX 69
SPRINGBORO, OH 45066
(513) 748-0390

8080 SIMULATOR

LISTING THREE (Continued from February)

```

move.l #hdbuf,d1
move.b d1,(a6)+                                ; Point host buffer to self.
ror.l #8,d1
move.b d1,(a6)+                                ;
ror.l #8,d1
move.b d1,(a6)+                                ;
move.b d1,(a6)+                                ;
move.l (sp)+,a6
move.l a0,-(sp)
suba.l targbase,a0
move.l a0,dmalink
move.l #$ff0000,a0
adda.l d0,a0
move.b rega,(a0)
move.l #hdbuf+12,a0
hdloop3 tst.b (a0)
beq hdloop3
move.b (a0),d1
move.l (sp)+,a0
move.b d1,-(a0)
move.l (sp)+,d3
jmp (return)                                     ; Wait for completion
                                                ; Fragile, but what do you want for $1.00?
                                                ; Grab the STATUS address in target space.
                                                ; And stash status for it.
                                                ; Return to simulation.

data
even
dmalink dc.l $50
hdbuf ds.b 16                                     ; Storage for current HDDMA command buffer.

end

```

End Listing Three

LISTING FOUR

```

*****
*   This file contains the mnemonic strings for the 8080 opcodes. *
*   These are used in tracing.                                         *
*****
globl mnops
data
page
even
mnops    dc.l mnnop00,mnlxib,mnstashb,mninxb,mninxr,mndcrb,mnmvib,mnrlca
dc.l mnnop08,mndadb,mnlaxdb,mndcxb,mninxrc,mndcrc,mnmvic,mnrrca
dc.l mnnop10,mnlxid,mnsta,mninxd,mninxr,mndcrd,mnmvid,mnral
dc.l mnnop18,mndadx,mnldadx,mndcxm,mninxre,mndcre,mnmvie,mnrr
dc.l mnnop20,mnlxih,mnshld,mninxh,mninxr,mndcrh,mnmvih,mnda
dc.l mnnop28,mndadh,mnlhid,mndcxh,mnirrl,mndcrl,mnmvil,mncma
dc.l mnnop30,mnlxis,mnsta,mninxs,mninxr,mndcrm,mnmvim,mnste
dc.l mnnop38,mndads,mnlida,mndcxm,mnirra,mndcra,mnmvia,mncmc
dc.l mnmovbb,mnmovbc,mnmovbd,mnmovbe,mnmovbh,mnmovbl,mnmovbm,mnmovba
dc.l mnmovcb,mnmovcc,mnmovcd,mnmovce,mnmovch,mnmovcl,mnmovcm,mnmovca
dc.l mnmovdb,mnmovdc,mnmovdd,mnmovde,mnmovdh,mnmovdl,mnmovdm,mnmovda
dc.l mnmoveb,mnmovec,mnmoved,mnmovee,mnmovew,mnmovel,mnmovem,mnmova
dc.l mnmovhb,mnmovhc,mnmovhd,mnmovhe,mnmovvh,mnmovhl,mnmovhm,mnmovha
dc.l mnmovlb,mnmovlc,mnmovld,mnmovle,mnmovlh,mnmovll,mnmovlm,mnmovla
dc.l mnmovmb,mnmovmc,mnmovmd,mnmovme,mnmovmh,mnmovml,mnhalt,mnmovma
dc.l mnmovab,mnmovac,mnmovad,mnmovae,mnmovah,mnmoval,mnmovam,mnmovaa
dc.l mnaddb,mnaddc,mnaddd,mnade,mnaddh,mnaddl,mnaddm,mnaddaa
dc.l mnadcb,mnadcc,mnadcd,mnadce,mnadch,mnadcl,mnadcm,mnadca
dc.l mnsubb,mnsubc,mnsubd,mnsube,mnsuh,mnsUBL,mnsUBM,mnsUBAA
dc.l mnsbbb,mnsbbc,mnsbbd,mnsbbe,mnsbbh,mnsbbL,mnsBBM,mnsBBA
dc.l mnandb,mnandc,mnandd,mnande,mnandh,mnndl,mnandm,mnanda
dc.l mnxrab,mnxrac,mnxrad,mnxrae,mnxrah,mnxral,mnxram,mnxraa
dc.l mnorab,mnorac,mnorad,mnorae,mnorah,mnoral,mnoram,mnoraa

```

```

dc.l mnccmpb,mnccmpc,mnccmpd,mnccmpe,mnccmph,mnccmpl,mnccmpam,mnccmpaa
dc.l mnrrnz,mnnpopb,mnjz,mnjmpa,mnncz,mnpuhb,mnadi,mnrsr0
dc.l mnrrz,mnret,mnjz,mnnpopCB,mncc,mncc,mnac1,mnaci,mnrsr8
dc.l mnrrc,mnnpopd,mnjc,mnout,mncc,mnpuhd,mnsui,mnrsr10
dc.l mnrrc,mnnpod9,mnjc,mnnc,mncc,mnnpodD,mnsbi,mnrsr18
dc.l mnrrpo,mnppoh,mnjpo,mnxth1,mncc,mnpuhd,mnani,mnrsr20
dc.l mnrrp,mnppop,mnjpo,mndi,mncc,mnpuhd,mnoria,mnrsr30
dc.l mnrrm,mnspshl,mnjm,mnei,mncc,mnnpodF,mnccpi,mnrsr38

page
*****
* Mnemonic Strings. The first character flags operands.
* Blank is nothing, A is an address, C is a constant.
*****
*****
```

mnnop00 dc.b " NOP\$"
mnlxib dc.b "ALXI B,\$"
mnstaxb dc.b " STAX B\$"
mninxb dc.b " INX B\$"
mninrb dc.b " INR B\$"
mndcrb dc.b " DCR B\$"
mnvvib dc.b "CMVI B,\$"
mnrlca dc.b " RLC\$"
mnnop08 dc.b " ILLEGAL FOR 8080\$"
mndadb dc.b " DAD B\$"
mnldaxb dc.b " LDAX B\$"
mndcxb dc.b " DCX B\$"
mninrc dc.b " INR C\$"
mndcrc dc.b " DCR C\$"
mnvvic dc.b "MVI C\$"
mnrrca dc.b " RRC\$"
mnnop10 dc.b " ILLEGAL FOR 8080\$"
mnlxid dc.b "ALXI D,\$"
mnstaxd dc.b " STAX D\$"
mninxd dc.b " INX D\$"
mninrd dc.b " INR D\$"
mndcrd dc.b " DCR D\$"
mnvvid dc.b "CMVI D,\$"
mnral dc.b " RAL\$"
mnnop18 dc.b " ILLEGAL FOR 8080\$"
mndadd dc.b " DAD D\$"
mnldaxd dc.b " LDAX D\$"
mndcxd dc.b " DCX D\$"
mninre dc.b " INR E\$"
mndcre dc.b " DCR E\$"
mnvvie dc.b "CMVI E,\$"
mnrrar dc.b " RAR\$"
mnnop20 dc.b " ILLEGAL FOR 8080\$"
mnlxih dc.b "ALXI H,\$"
mnshld dc.b "ASHLD \$"
mninxh dc.b " INX H\$"
mninrh dc.b " INR H\$"
mndcrh dc.b " DCR H\$"
mnvvih dc.b "CMVI H,\$"
mnada dc.b " DAAS\$"
mnnop28 dc.b " ILLEGAL FOR 8080\$"
mndadh dc.b " DAD H\$"
mnlhld dc.b "ALHLD \$"
mndcxh dc.b " DCX H\$"
mninrl dc.b " INR L\$"
mndcrl dc.b " DCR L\$"
mnvvil dc.b "CMVI L,\$"
mncma dc.b " CMAS"
mnnop30 dc.b " ILLEGAL FOR 8080\$"
mnlxis dc.b "ALXI S,\$"
mnsta dc.b "ASTA \$"
mninxs dc.b " INX S\$"
mninrm dc.b " INR M\$"
mndcrm dc.b " DCR M\$"
mnvvim dc.b "CMVI M,\$"
mnstc dc.b " STCS\$"
mnnop38 dc.b " ILLEGAL FOR 8080\$"
mndads dc.b " DAD S\$"
mnlda dc.b "ALDA \$"
mndcxs dc.b " DCX S\$"
mninra dc.b " INR A\$"

(Continued on next page)



Thinking about C?

Stop Thinking-
Start Programming Today!

SPECIAL INTRODUCTORY OFFER!
C' Prime, Personal Computing and C,
Plus Apprentice C.
A \$169 value only

\$99

NEW FROM MANX AZTEC!

C' Prime

~~\$99~~ **\$79**

Never has C been easier to learn. **Manx Aztec** is now offering a complete C system called **C' Prime** at an exceptionally low price. This powerful system includes a Compiler, Linker, Assembler, Editor, Libraries and Object Librarian. **C PRIME** supports a host of third-party software.

C Apprentice ~~\$49.95~~ **\$39.95**

Learn C quickly with this complete, easy-to-use C language interpreter. **Apprentice C** includes a complete one-step compiler that executes with lightning speed, an editor, and a run-time system.

NEW FROM ASHTON-TATE!

Personal Computing and C

A detailed, easy-to-understand guide to C programming prepared especially by Ashton-Tate for use with the new **Aztec C' Prime**. Includes chapters on C programming basics, function libraries, data handling, and advanced features, plus a complete money management demonstration program.

To order or for information call:

TECWARE
1-800-TEC-WARE
(In NJ call 201-530-6307)



UNIX is a registered TM of Bell Laboratories. dBase TM Ashton-Tate, Inc. MANX AZTEC, C PRIME, Apprentice C, TM Manx Software Systems, Inc.

Circle no. 222 on reader service card.

8080 SIMULATOR

LISTING FOUR (Continued from February)

mnndra dc.b " DCR A\$" mnrvia dc.b " CMVI A,\$" mncmc dc.b " CMCS\$" mnmovbl dc.b " MOV B,B\$" mnmovbc dc.b " MOV B,C\$" mnmovbd dc.b " MOV B,D\$" mnmovbe dc.b " MOV B,E\$" mnmovbh dc.b " MOV B,H\$" mnmovbl dc.b " MOV B,L\$" mnmoym dc.b " MOV B,M\$" mnmovba dc.b " MOV B,A\$" mnmovcb dc.b " MOV C,B\$" mnmovcc dc.b " MOV C,C\$" mnmovecd dc.b " MOV C,D\$" mnmovec dc.b " MOV C,E\$" mnrovch dc.b " MOV C,H\$" mnmovcl dc.b " MOV C,L\$" mnmovcm dc.b " MOV C,M\$" mnmovca dc.b " MOV C,A\$" mnmovdb dc.b " MOV D,B\$" mnmovdc dc.b " MOV D,C\$" mnmovdd dc.b " MOV D,D\$" mnmove dc.b " MOV D,E\$" mnmovdh dc.b " MOV D,H\$" mnmovdl dc.b " MOV D,L\$" mnmovdm dc.b " MOV D,M\$" mnmovda dc.b " MOV D,A\$" mnmoveb dc.b " MOV E,B\$" mnmovec dc.b " MOV E,C\$" mnmoved dc.b " MOV E,D\$" nnmovee dc.b " MOV E,E\$" nnmoveh dc.b " MOV E,H\$" nnmovef dc.b " MOV E,L\$" nnmovem dc.b " MOV E,M\$" nnmovea dc.b " MOV E,A\$" nnmovhb dc.b " MOV H,B\$" nnmovhc dc.b " MOV H,C\$" nnmovhd dc.b " MOV H,D\$" nnmovhe dc.b " MOV H,E\$" nnmovhh dc.b " MOV H,H\$" nnmovhl dc.b " MOV H,L\$" nnmovhm dc.b " MOV H,M\$" nnmovha dc.b " MOV H,A\$" nnmovlb dc.b " MOV L,B\$" nnmovlc dc.b " MOV L,C\$" nnmovld dc.b " MOV L,D\$" nnmove dc.b " MOV L,E\$" nnmovlh dc.b " MOV L,H\$" nnmovl1 dc.b " MOV L,L\$" nnmovlm dc.b " MOV L,M\$" nnmovla dc.b " MOV L,A\$" nnmovmb dc.b " MOV M,B\$" nnmovmc dc.b " MOV M,C\$" nnmovmd dc.b " MOV M,D\$" nnmovme dc.b " MOV M,E\$" nnmovmh dc.b " MOV M,H\$" nnmovml dc.b " MOV M,L\$" mnhalt dc.b " HALT\$" mnmovma dc.b " MOV M,A\$" nnmovab dc.b " MOV A,B\$" nnmovac dc.b " MOV A,C\$" nnmovad dc.b " MOV A,D\$" nnmoveae dc.b " MOV A,E\$" nnmovah dc.b " MOV A,H\$" nnmoval dc.b " MOV A,L\$" nnmovam dc.b " MOV A,M\$" nnmovaa dc.b " MOV A,A\$" mnaddb dc.b " ADD B\$" mnaddc dc.b " ADD C\$" mnaddd dc.b " ADD D\$" mnadde dc.b " ADD E\$" mnaddh dc.b " ADD H\$" mnaddl dc.b " ADD L\$"	mnaddm dc.b " ADD M\$" mnaddaa dc.b " ADD A\$" mnadcb dc.b " ADC B\$" mnadcc dc.b " ADC C\$" mnadcd dc.b " ADC D\$" mnadce dc.b " ADC E\$" mnadch dc.b " ADC H\$" mnadcl dc.b " ADC L\$" mnadcm dc.b " ADC M\$" mnadca dc.b " ADC A\$" mnsubb dc.b " SUB B\$" mnsubc dc.b " SUB C\$" mnsubd dc.b " SUB D\$" mnsube dc.b " SUB E\$" mnsubh dc.b " SUB H\$" mnsubl dc.b " SUB L\$" mnsubm dc.b " SUB M\$" mnsubaa dc.b " SUB A\$" mnsubbb dc.b " SBB B\$" mnsubbc dc.b " SBB C\$" mnsubbd dc.b " SBB D\$" mnsubbe dc.b " SBB E\$" mnsubbh dc.b " SBB H\$" mnsubbl dc.b " SBB L\$" mnsubbm dc.b " SBB M\$" mnsubba dc.b " SBB A\$" mnandb dc.b " ANA B\$" mnandc dc.b " ANA C\$" mnandd dc.b " ANA D\$" mnande dc.b " ANA E\$" mnandh dc.b " ANA H\$" mnndl dc.b " ANA L\$" mnandm dc.b " ANA M\$" mnanda dc.b " ANA A\$" mnxrab dc.b " XRA B\$" mnxrac dc.b " XRA C\$" mnxrard dc.b " XRA D\$" mnxrae dc.b " XRA E\$" mnxrah dc.b " XRA H\$" mnxral dc.b " XRA L\$" mnxram dc.b " XRA M\$" mnxraa dc.b " XRA A\$" mnorah dc.b " ORA H\$" mnoral dc.b " ORA L\$" mnoran dc.b " ORA M\$" mnoraa dc.b " ORA A\$" mncmpb dc.b " CMP B\$" mncmpc dc.b " CMP C\$" mncmpd dc.b " CMP D\$" mncmpc dc.b " CMP E\$" mncmpf dc.b " CMP H\$" mncmp1 dc.b " CMP L\$" mncmpam dc.b " CMP M\$" mnmpaa dc.b " CMP A\$" mnrrz dc.b " RNZ\$" mnret dc.b " RET\$" mnpopb dc.b " POP B\$" mnjnz dc.b " AJNZ \$" mnjmpa dc.b " AJMP \$" mncnz dc.b " ACNZ \$" mnpushb dc.b " PUSH B\$" mnadi dc.b " CADI \$" mnrst0 dc.b " RST 0\$" mnrz dc.b " RZ\$" mnjz dc.b " AJZ \$" mnnopCB dc.b " ILLEGAL FOR 8080\$" mncz dc.b " ACZ \$" mnccall dc.b " ACALL \$" mnaci dc.b " CACI \$"	mnrst8 dc.b " RST 8\$" mnrcn dc.b " RNC\$" mnpopd dc.b " POP D\$" mnjnc dc.b " AJNC \$" mnout dc.b " COUT \$" mncnc dc.b " ACNC \$" mnpushd dc.b " PUSH D\$" mn sui dc.b " CSUI \$" mnrst10 dc.b " RST 10\$" mnrc dc.b " RC\$" mnnopD9 dc.b " ILLEGAL FOR 8080\$" mnjc dc.b " AJC \$" mnin dc.b " CIN \$" mncc dc.b " ACC \$" mnnopDD dc.b " ILLEGAL FOR 8080\$" mnsbi dc.b " CSBI \$" mnrst18 dc.b " RST 18\$" mnropo dc.b " RPO\$" mnpoph dc.b " POP H\$" mnjpo dc.b " AJPO \$" mnxthl dc.b " XTHL\$" mncpo dc.b " ACPO \$" mnpushh dc.b " PUSH H\$" mnani dc.b " CANI \$" mnrst20 dc.b " RST 20\$" mnrpe dc.b " RPE\$" mnpchl dc.b " PCHL\$" mnjpe dc.b " AJPE \$" mnxchg dc.b " XCHG\$" mncpe dc.b " ACPE \$" mnpred dc.b " ILLEGAL FOR 8080\$" mnxri dc.b " CXRI \$" mnrst28 dc.b " RST 28\$" mnrrp dc.b " RPS\$" mnpopp dc.b " POP P\$" mnjpr dc.b " AJP \$" mndi dc.b " DI\$" mncp dc.b " ACP \$" mnpushp dc.b " PUSH P\$" mnoria dc.b " CORI \$" mnrst30 dc.b " RST 30\$" mnrm dc.b " RMS\$" mnspfh dc.b " SPHL\$" mnjm dc.b " AJM \$" mnei dc.b " EI\$" mncm dc.b " ACM \$" mnnopFD dc.b " ILLEGAL FOR 8080\$" mncri dc.b " CCPI \$" mnrst38 dc.b " RST 38\$" .end
---	--	--

End Listings



The easy life

Don't let anyone tell you that there is an easier or faster way of producing C source code than using **PRO-C** —— There Isn't.

PRO-C generates fully optimized, perfectly commented and professionally documented programs to your requirements in minutes.

PRO-C consists of the following modules:

PRO-C 'Painters'

- Allows manipulation of prompts, data fields, free form text, and box drawing
- Allows for choice of video display attributes
- Provides you with the latest in WYSIWYG technology (What You See Is What You Get)

MENU Generator

- Utilizes the 'painters' to design menus
- Assists the integration of generated programs from other modules
- Creates professionally designed, menu-driven systems

REPORT Generator

- Generates multi-file reports utilizing the 'painters'
- Permits the following report layout types:
 - column style
 - mailshots
 - mailing labels
 - preformatted stationary
- Reduces time spent designing reports

SCREEN Generator

- Generates multi-file data entry programs utilizing the 'painters'
- Permits the generation of the following functions in any combination:
 - Addition of records
 - Modification of records
 - Deletion of records
 - Inquiry of records
- Enhances visual professionalism and increases productivity

UPDATE Generator

- Generates multi-file update programs
- Provides sort/merge and data manipulation capabilities

RECORD Definition

- Allows for record definition of any data files required within a proposed system
- Provides full range of 'C' data types
- Links files relationally, independent of the ISAM file handler or I/O protocol used

File Handling & On-line Help

PRO-C will interface with any ISAM handler and support industry names such as Btrieve and C-ISAM.

PRO-C comes with ½ megabyte of context sensitive on-line help.

PRO-C permits the incorporation of unlimited on-line help in any program.

PRO-C generated programs require No Run Time System or License.

The Professionals' C-Code Generator

REQUIREMENTS

- MS-DOS / PC-DOS V2.0 or later
- 256K RAM
- Hard Disk & 5 ¼" Flexible Disk Drive
- One of the following C Compilers:
Microsoft C V3.0, Lattice C V2.15,
Computer Innovations C86 V2.3F,
DeSmet C V2.4

PRO-C WORKBENCH

The source code to the library routines used throughout **PRO-C**. A valuable tool for the programmer in assisting in fine tuning applications to personal preference. Also facilitates 'porting' to various machines.

Circle no. 270 on reader service card.



Yes, please send me the tools for the EASY LIFE!

- PRO-C @ \$195**
 PRO-C WORKBENCH @ \$195
 PRO-C & WORKBENCH @ \$295

Visa MasterCard Check Money Order

Enclosed Amount _____

Indiana residents add 5% sales tax. Postage and Handling - In U.S. \$4.00, Outside U.S. \$10.00.

Name _____ Company _____

Address _____ Phone _____

City _____ State _____ Zip _____

VISA or MC # _____ Exp. Date _____

ORDERS

Mail or Phone Orders to:

U.S.A.
MAJIC Software Inc.
224 South Salisbury Suite C4
West Lafayette, IN 47906
(317) 743-0610

ENGLAND
C.C. Limited
Beacon House, Pyrford Rd.
Surrey KT14 6LD
09323 55172

THE RIGHT TO ASSEMBLE

NS32000 Square Roots

The square root routine for the 68000 by Jim Cathey (16-Bit Software Toolbox, May 1985) is an example of the looping and bit-shifting routines that must be used on a microprocessor with limited arithmetic ability (i.e., 8- or 16-bit registers and only addition and subtraction). However, this kind of bit-level thinking seems unnecessary and even undesirable with a microprocessor such as the NS320XX or the 68000. A programmer would be better off attacking such problems as computing square roots from a higher level.

The NS320XX group is made by National Semiconductor (my NS16032 was an earlier version) and the 68000 by Motorola: They can perform 8-, 16-, and 32-bit addition, subtraction, multiplication, division, and modulo operations on data in registers and memory. The NS320XX, in particular, was clearly designed with high-level language compilation in mind. Although it does have bit-level capabilities, it actually performs better at a higher level.

I have described a routine for calculating square roots and have presented it in both C-language program form (Listing One, page 106) and assembly language for the NS320XX processor (Listing Two, page 106). Compiling, assembling, and linking takes about 1½ minutes on a system with relatively slow floppy-disk access. I hope that one of these presentations will be helpful to you. I often find it frustrating to try to understand an interesting programming approach to a problem when it is only general-

ber and, of course, the square of the square root equals the number. When dealing with integers, however, the situation is somewhat different. The square root of an integer is the smallest of the two most nearly equal factors of that number. For example, the two most nearly equal integer factors of 255 are 15 and 17, and the square root of 255 is 15. The integer square root when squared may not equal the number.

This routine has two stages. The first is to find two roughly equal factors of the number, *guess1* and *guess2*, in the quickest possible way. Initially, *guess1* is set to 1 and *guess2* is set to the number for which the square is desired. *Guess1* is then multiplied and *guess2* is divided by two until their values are approximately equal. An average of the two guesses is calculated as the trial square root, and we move to the testing stage. In all cases, multiplication and division by two are done by the faster methods of left and right shifting.

This trial square root now becomes *guess1* and is tested by dividing it into the number, and the quotient becomes *guess2*. An error term is calculated by subtracting *guess2* from *guess1*. If this error is negative or zero, *guess1* is taken as the square root. If this error is positive (the trial square root is too high), a new trial square root (the mean of the two guesses) is computed, and the above division test is repeated until the error becomes zero or negative.

The trial square root from the factoring stage is biased high, which is advantageous because it leads to downward adjustment only of the trial square root. This is good because when the correct square root of an integer number is divided into the number it may yield a quotient greater than the square root. For example, consider the integer square root of 24. [The first trial square root would be 5 (the mean of 4 and 6), 24 divided

by 5 is 4, because the error (5–4) is +1, 4 is not the correct root.] The next trial square root would be 5 plus 4 divided by 2, or 4. Because 24 divided by 4 equals 5, an error of –1, the integer square root of 24 is 4.

Although you might be suspicious of it, the above description should make sense, particularly if you try it out with paper and pencil. Though the bit-shifting routines are virtually opaque, this routine does work accurately and fast.

Test runs computing the square root of integers from 0 to 60,000, as shown in Listing Three, page 106, indicate that on the average 7.0 shifts (excluding the test) and 1.8 divisions per square root were carried out; 25 percent of the times only one division was required, and the maximum was four. Running the full 60,000 square root program on a 6-MHz NS16032 CPU took 27 seconds and 17 seconds without printing or counting shifts and divisions. The program only prints the square root when it changes from the previously calculated one.

Running the program without printing, counting, or square root calculations took 6 seconds; therefore, the 60,000 square roots took 11 seconds (17–6) to calculate. Thus, the average time per square root is a rather respectable 183 microseconds. The assembly-language routine was compiled and could be hand optimized. A couple of branches could be eliminated and the error term would not have to be stored. Is it worth the programming time? Not to me.

Listing Four, page 106, shows Cathey's bit-shifting version as coded for the NS320XX. Although it contains fewer instructions, it actually is considerably slower than the dividing routine. When timed in the same fashion as the shift and divide routine, it takes an average of 413 microseconds.

I believe that the NS320XX is the

by Richard A. Campbell

ly described or when it is presented in a language I am vaguely familiar with.

The square root of a real number is one of two equal factors of that num-

most powerful microprocessor around. Its generally symmetrical instruction set and many addressing modes make it relatively easy to write systems software for. After all, there is more to programming than calculating square roots! But its apparently poor showing for the bit-shifting routine, relative to the M68000 (413 vs. 200 to 250 microseconds), deserves some comments. If my two-year-old NS16032 were running 8 MHz, then the difference would be somewhat less (310 vs. 200 to 250 microseconds). Also, the M68000 times were estimated from instruction-execution clock cycles, not with an actual test program; calculating instruction-execution times "in vivo" for sophisticated microprocessors such as the NS320XX or the M68000 is not an easy task.

In any event, this short test and bit-shifting algorithm does hit the NS320XX below the belt in two ways. First, the NS320XX has a single lookahead instruction queue that may actually cost time when executing short loops or frequent branches; this short, 14-instruction routine has to loop 16 times and may branch up to 16 times. Second, it does not have a simple bit-setting instruction for the efficient movement of single bits from a data or flag register to another register. The first problem is probably inherent in its design, but the second is a less serious design or documentation oversight.

This little exercise has further convinced me that you are better off programming at a level where the logic of a program is evident. This is particularly true with a modern microprocessor such as the NS320XX. I would urge anyone working with such a microprocessor to forget about the bit-level tricks necessary with 8-bit processors and take a fresh and rational approach to programming.

DDJ

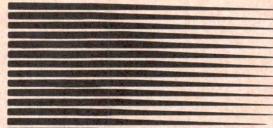
(Listings begin on page 106)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 7.

PROGRAMMING TOOLS

DEMO DISK
\$10.00
(credited towards
future purchase)



Version (4.0)

C-Plus (For IBM Microsoft C)

Turbo-Plus (For Turbo-Pascal)

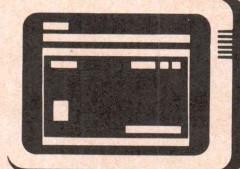
Pascal-Plus (For IBM-Microsoft Pascal)

Compile 50 times faster.

Plus Software is a library of procedures crafted in assembly language and designed to enhance Turbo-Pascal™ and IBM-Microsoft Pascal. No royalties are required for applications developed using Plus Software. These procedures are extremely fast and combine automatically with your programs. The benefit to you is faster development and compilation time and smaller, faster application programs. Features include:

- Multiple input/output field display map generation with edit masks
- Instant text write
- Instant attribute write (the fancy cursor maker)
- Display chunk retrieval
- Graphics text write (any size or position)
- Pop-up any time reference guide
- Pull Down Menu Maker
- Snow removal
- RamWindow
- Advanced keyboard control
- Multiple screen snapshots and restores
- File handle I/O
- Variable Length
- Sample programs with source code
- Printed manual

\$59.95 (Requires IBM PC and compatibles)



Screen Genie

The ultimate screen editor at any price.

- Creates and edits screens for programmers and non-programmers
- Resident pop-up facility for creating your own pop-ups
- Full typematic and cursor control. Insert, overwrite in two dimensions
- Paints/unpaints, draws/undraws in any direction. Easy to use selection menus
- Copy/move lines or window pieces with forgiving adjustment feature.
- Save and edit any size windows.
- SCREEN GRABBER lets you save and edit any screen from any application
- CREATES FIELDS AND GENERATES CODE FOR PROGRAMMERS
- Creates screens as executable files for non-programmers
- Creates load files (for any language), .obj files, .com external procedures
- Non-copy protected

\$69.95 (Requires IBM PC and Compatibles)



Turbo-Spawn

(For Turbo-Pascal)

\$39.95

(Requires MS PC DOS)

Turbo Spawn lets you execute any size program including another Turbo Pascal program or Dos commands without leaving current program and continue with next instruction. **Breaks the 64K barrier.**

Nostradamus Inc.

Software
sourcery

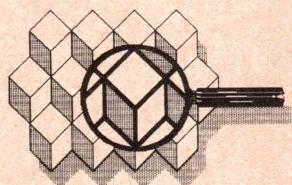
Please send me:		<input type="checkbox"/> Check	<input type="checkbox"/> Money Order	Name _____
<input type="checkbox"/>	C-Plus	\$59.95	<input type="checkbox"/>	Visa/MasterCard/Amex
<input type="checkbox"/>	Turbo-Plus	\$59.95	NO. _____	Address _____
<input type="checkbox"/>	Pascal-Plus	\$59.95		
<input type="checkbox"/>	Screen Genie	\$69.95		
<input type="checkbox"/>	Turbo-Spawn	\$39.95		
<input type="checkbox"/>	Demo Disk	\$10.00	Exp. Date _____	City/State/Zip _____
Total amount		U.S. shipping and handling included _____ Outside U.S. pay postage _____		



Nostradamus Inc. / 5320 South 900 East, Suite 110
SLC, Utah 84117 / Order by phone (801) 261-0769

Circle no. 251 on reader service card.

OF INTEREST



To write a program of moderate size that works as soon as it's compiled is a programming feat. Programmers understand the need for debugging tools in the same way that an engineer understands the need for an oscilloscope. Unfortunately, programmers generally have available only the most rudimentary debugging facilities.

To help programmers out of the morass of loads, stores, jumps, and bit-fondling, Meridian Software Systems has developed an interactive system-independent source-level debugger for use with its Pascal compiler. It supports conditional breakpoints, subprogram traces, single-stepping, and full Pascal variable reference syntax.

TurboRef, a software tool package for Pascal users, is available from Graccon Services. TurboRef lists each variable and constant reference in a user's program. The list includes the line number and type of use for each reference. It can read multiple source files and process "include" files. A program listing can also be created. The product runs on an IBM PC with 128K memory and requires PC DOS 2.00. The price is \$49.95.

The SPS C/Atlas compilation system from Software Products & Services is modularized to enable the installation of a total system or only those elements that are needed to round

out or upgrade an existing capability. The system consists of a front-end compiler, syntax-oriented editor, code-generator compiler, TDL processor, adaptation list generator, device and switch allocation system, test and debug package, and consistency checker.

ProMod, an integrated systems engineering environment from Promod Inc., supports the complete software life cycle on PCs as well as minicomputer systems. Versions are available for IBM PCs having 512K of memory and 10-Mb hard-disk storage and DEC VAX systems operating under VMS.

Version 1.0 of P-TRAL is a language translator that enables programmers to convert Applesoft BASIC programs to Pascal. It reads the BASIC source program from disk and generates the equivalent Pascal source code. System requirements include: 64K to 128K memory, 80-column display, two disk drives, and Apple DOS 3.3/Apple Pascal 1.1, 1.2, 1.3. P-TRAL is available from Woodchuck Industries.

Cipherlink Corp. has introduced Calculations Peripheral, a software package that, when used with Any bridge software, translates data between incompatible computers. The Any data bridge resides on an independent computer connected to the source and target computers through a terminal port. It acts as a bridge between applications, extracting a desired data field from a source application, transferring it to an intermediate database, and then automatically rekeying the data into the appropriate field in the target ap-

plications. Calculations Peripheral will run on any PC DOS or Unix-based computer.

A three-part software package for analyzing and managing the IBM PC Network is available from Bickley Utilities. Net Set prompts the user interactively for information concerning the network and generates definitions for the complex parameters involved in the Net Start command. Net Stat is the on-line network statistics-gathering module.

The Omni 64 from Oliver Advanced Engineering is plug-compatible with over 300 computers and operating systems. It has 65,536 bytes of RAM, expandable to 262,144 bytes, and an unlimited firmware database. The product comes with eight logical buffers that can be configured to a different depth (to 256K words) and width (to 64 bits) to correspond with up to eight different programmable devices. The Omni 64 uses a 6-MHz 280B processor to synchronize the processor to the waveforms being produced.

The Local Applications Bus, LAB 40, is a microcomputer-to-peripheral interface and a hardware development system available from Computer Continuum. LAB 40 is a structured parallel port that takes 16K memory or I/O locations in the host computer. It can drive up to 64 8-bit input ports and 64 8-bit output ports.

AT SpeedFixer is a software utility package, available from Dynamical Systems. It features a memory-resident disk utility designed to prevent floppy disk drive errors and a keyboard utility that

increases the speed. The price is \$24.95.

C. Itoh Products has introduced two high-resolution RGB monitors. The CM1000 features both composite and RGB capability and a full-range audio speaker. In composite mode, it provides a 320 × 240 resolution with a 4-MHz bandwidth; in RGB, the resolution jumps to 640 × 240 with a 15-MHz bandwidth. The CM2000 features a nonglare black screen and is designed to be plug-compatible with the IBM PC, XT, and AT as well as the Apple IIe series.

A four-channel Multibus single-board computer designed for sophisticated data communication processing is available from SBE Inc. The COM-4 has two 10-MHz user-programmable DMA controllers that provide each of the four on-board serial ports with two channels of DMA. These channels transmit and receive blocks of data to and from on-board RAM memory. The on-board memory is dual-ported, allowing data to be accessed from either the local bus or Multibus.

Communication Machinery Corp. has announced the ENP-60, a high-performance intelligent front-end processor on a single board. The board runs the communication protocols required for the IBM PC/AT to communicate to other devices on Ethernet. The ENP-60 also runs TCP/IP software as well as the Fusion higher-level protocol software for XNS. MS DOS and Xenix are also supported.

A Virtual RAM (VRAM) card, designed to increase the capabilities and performance of the Waterloo Port Network Operating Sys-

tem, is available from Waterloo Microsystems. Equipped with a VRAM card, a port workstation can run 640K DOS applications, concurrently access network services, send mail, and perform terminal emulation and other multi-tasking applications. The VRAM card is a full-length board with 512K of RAM, expandable to 1 Mb. It operates with the IBM PC, XT, AT, and compatibles at speeds up to 8 MHz. The price is \$695 (\$995 in Canada).

The GL Serial Card from Microtek provides a two-way serial interface to allow the IBM PC/XT to communicate with a variety of serial peripheral devices. It is functionally equivalent to the IBM asynchronous communication adapter and supports all software designed to run with this

card on the IBM PC/XT.

Quepro Version 1.5, enables managers to analyze data files using programs such as Lotus 1-2-3, dBASE III, and MailMerge. Available from Bytel Corp., the new version permits users to generate ad hoc queries and reports without the need for programmer assistance. It can be used as is with files generated using the Cogen COBOL program generator or customized by the developer of any COBOL program. Version 1.5 is priced at \$595 for MS DOS systems, \$785 to \$1,550 for most Unix-based systems, and \$1,550 to \$9,500 for NCR and large Unix systems.

Ampère has developed a 68000-based portable computer called the WS-1. It includes up to 512K of RAM, an 80-character-25-line LCD

(200 × 480 = dot bit-mapped graphics also supported), built-in speaker phone, auto-dial 300-baud modem, programmable voice/data microcassette, two RS-232 ports, and disk I/O extension bus. WS-1 also uses a multitasking operating system called BIG DOX. WS-1 is available from Workspace Computer.

Boston Business Computing has introduced the PC/EDT, a PC implementation of BAX EDT, the de facto Digital mainframe editor. It features multiple buffers, undelete, keyboard redefinition, command macros, multiple file access, cut and paste, screen and/or keyboard command modes, command files, and environmental variables. It supports DOS 2.0 and above as well as the full DOS directory struc-

ture. The program, priced at \$220, requires a single disk drive and 128K memory.

A down-size 201C modem that fits under a standard telephone is available from Emerald Technology Group. The 201C features digital signal processing under microprocessor control. It is designed for use on switched lines and includes circuitry for auto-answer operation. The modem, priced at \$685, also incorporates several diagnostic test functions including loopback, random-bit test patterns, and self-test.

Turn-On is an intelligent power controller from Dynatech that provides unattended remote access plus power protection. It features automatic log-on three-phase security: user

Parlez-vous Pascal?

Mais oui.

After you've digested this, you will.

Introducing "Exploring Pascal: A Compiler For Beginners."

The fastest way on the planet to learn and use Pascal.

So you can put some of your programming ideas to work, to accomplish your specialized needs. And have the power of a pseudo-compiler, too.

It's fast because we make it as easy as possible.

Exploring Pascal is a unique, multimedia book and disk learning system.

The manual is very easy to read. Because you don't want to have to struggle with the

English language to learn a new computer language.

The software tutorials are interactive, and the exercises are computer graded. So you know how you're doing, every step of the way.

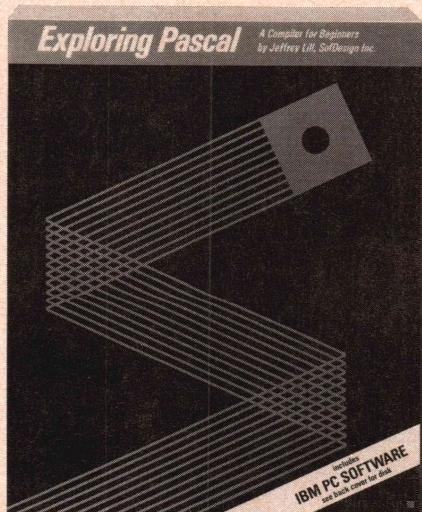
And there are animated demonstrations of important concepts.

400 screens of help in all.

To order, for the name of your nearest dealer, or for more information, just call the Ashton-Tate Publishing Group at 800-437-4329, Ext. 248.

If you're computer literate and are ready to step up to computer programming, it's really très bon.

Trademark/owner: Ashton-Tate. © 1986 Ashton-Tate. All rights reserved. Specifications subject to change without notice.



Circle no. 248 on reader service card.

OF INTEREST

(Continued from page 125)

ID, password protection, and optional dial-back confirmation.

IBM PC

The ConCur 400, an advanced adapter board that transforms single-user programs into multiterminal programs, is available from Vaxon. The board provides noninterlaced video for flicker-free display and 720×348 monochrome high-resolution graphics with an IBM monitor or 640×400 with a Princeton SR-12 or compatible monitor. It runs on the IBM PC, XT, AT, and compatibles.

Expert Edge is an expert system builder that can be used to develop interactive knowledge-based programs for the IBM PC. It uses a deductive reasoning (backward chaining) inference engine to generate expert advice or to solve a problem requiring expert knowledge. Starting from rules entered by the expert, Expert Edge builds an interactive program, called an advisor, that can then be referred to by any PC user. These rules can incorporate calculations, equations, logical reasoning, judgment, facts, and uncertainties. Features such as "True" and "Why" can follow the chain of logic being used forward or backward as well as show the builder exactly how any conclusion has been reached. The package is available from Human Edge Software and sells for \$795.

A software library of more than 8,800 IBM PC programs on a single CD-ROM disk is available from Reference Technology. It includes a selection of word processors, editors, database management sys-

tems, spreadsheets, financial and business applications, communication programs, math/statistical packages, education and music programs, and games. The price is \$850.

The Coretape system, a tape backup for the IBM PC, XT, AT, and compatibles, comes in both external and internal versions. Each offers 60 Mb of data storage. Coretape's data transfer rate is 90,000 bytes per second with the tape streaming at 90 inches per second. The system is available from CORE International.

UX Software created UX-Basic to service the IBM PC and compatibles running either the PC DOS or MS DOS operating systems. UX-Basic is compatible with the earlier model, Version 2.0. Among its features are structured code; modular programming; and sequential, direct, and ISAM files.

LDR Systems ISONET package provides independent implementation of ISO standards for open systems interconnection. The package can interconnect local area networks using X.25 protocols. It is available for the IBM PC and a number of compatibles including the Olivetti M24 and the Ericsson PC. It also runs on the Burroughs B20 range, the DEC VAX-II, the ACT Apricot, and Sirvius.

The Capture image digitizer board for IBM PC, XT, AT, and compatibles is available from Genoa Systems Corp. The board allows users to capture an image from a standard RS-170 video input, such as a camera. The captured image frame is then stored in on-board memory. The acquired image can also be simultaneously displayed on any composite graphics or TV monitor or printed on a standard graphics printer. The bit-mapped display

supports high-screen resolution of 512×512 pixels. A frame memory of $256K \times 8$ bits allows graphics overlay, requiring an address space of only 64K.

C, Unix, and Xenix

Computer Innovations announced a C compiler for use with Digital Equipment Corp.'s VAX systems (running under VMS) to develop ROM code for the Intel family of microprocessors. The C86 ROM-C compiler consists of a four-pass C compiler that emits Intel-8086 family object or assembler code, with code-generation options for 80186 and 80286 processors and 8087-80287 math coprocessors. Full library source code and an object module translator program are also included.

Information Processing Techniques Corp. announced revisions of two C language development tools: Tracer for C debugging and C68K C Cross Compiler for the Motorola 68000 environment. Revision 3.0 of Tracer features interactive run-time debugging. Revision 2.0 of C68K includes the 68010 and 68020 as target processors.

Unibol from Software Ireland Ltd. is a commercial programming language for Unix operating systems. It allows existing DIBOL-83 programs to run under Unix System III, System V, and those look-alikes supporting the full range of system calls and providing the C standard I/O library. The package consists of a compiler, a run-time interpreter, a symbolic debugger, and a library of external utility subroutines.

UniPress Software announced software products for the IBM PC/AT running the Xenix operating

system. UniPress II's integrated office automation package offers word processing, spreadsheet, and relational database facilities. Softkeys display available editing options in all the components. UniPress EMACS is a multiwindow full-screen editor that includes high-level programming aids and is extensible by user-defined macros and the built-in compiler MLISP programming language. Q-Calc interfaces with the Unix environment through pipes and filters and allows users to create command scripts for spreadsheet routines.

Reference Map

Bickley Utilities (The), 101 Tyler Ct., Ambler, PA 19002; (215) 628-3750. Reader Service Number 16.

Boston Business Computing, 360 Merrimack St., Lawrence, MA 01843; (617) 683-7920. Reader Service Number 17.

Bytel Corp., 1029 Solano Ave., Berkeley, CA 94706; (415) 527-1157. Reader Service Number 18.

Cipherlink Corp., 3807 Wilshire Blvd., 7th Floor, Los Angeles, CA 90010; (213) 387-5371. Reader Service Number 19.

C. Itoh Digital Products, 19750 S. Vermont Ave., Ste. 220, Torrance, CA 90502; (213) 327-2110. Reader Service Number 20.

Communication Machinery Corp., 1421 State St., Santa Barbara, CA 93101; (805) 963-9471. Reader Service Number 21.

Computer Continuum, 75 Southgate Ave., Ste. 6, Daly City, CA 94015; (415) 755-1978. Reader Service Number 22.

Computer Innovations, 980 Shrewsbury Ave., Tinton Falls, NJ 07724; (800) 922-0169. Reader Service Number 23.

CORE International, 7171 N.



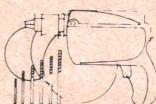
Federal Hwy., Boca Raton, FL 33431; (305) 997-6044. Reader Service Number 24. Dynamical Systems, 2511 Fulton St., Berkeley, CA 94704; (800) 227-2400. Reader Service Number 25. Dynatech Computer Power, 4744 Scotts Valley Dr., Scotts Valley, CA 95066; (408) 438-5760. Reader Service Number 26. Emerald Technology Group, 16701 116th NE, Bellevue, WA 98004; (206) 462-8200. Reader Service Number 27. Genoa Systems, 73 E. Trimble Rd., San Jose, CA 95131; (408) 945-9720. Reader Service Number 28. Gracon Services, 4632 Okemos Rd., Okemos, MI 48864; (517) 349-4900. Reader Service Number 29. Human Edge Software, 2445 Faber Pl., Palo Alto, CA 94303; (415) 493-1593.

Reader Service Number 30. Information Processing Techniques, 1096 E. Meadow Circle, Palo Alto, CA 94303; (415) 494-7500. Reader Service Number 31. LDR Systems (British Information Systems), 845 Third Ave., New York, NY 10022; (212) 752-8400. Reader Service Number 32. Meridian Software Systems, 130 McCormick Ave., Ste. 102, Costa Mesa, CA 92626; (714) 754-6631. Reader Service Number 33. Microtek, 5555 Magnation, Ste. H, San Diego, CA 92111; (619) 569-0900. Reader Service Number 34. Oliver Advanced Engineering, 676 W. Wilson Ave., Glendale, CA 91203; (213) 240-0080. Reader Service Number 35. Promod Inc., 22981 Alcalde Dr., Laguna Hills, CA 92653; (714) 855-3046. Reader Ser-

vice Number 36. Reference Technology, 1832 N. 55th St., Boulder, CO 80301; (800) 225-2749. Reader Service Number 37. SBE Inc., 2400 Bisso Ln., Concord, CA 94520; (800) 221-6458. Reader Service Number 38. Software Ireland Ltd., 27665 Forbes Rd., Laguna Niguel, CA 92677; (714) 582-0233. Reader Service Number 39. Software Products & Services, 14 E. 38th St., 14th Floor, New York, NY 10016; (212) 686-3790. Reader Service Number 40. UniPress Software, 2025 Lincoln Hwy., Edison, NJ 08817; (201) 985-8000. Reader Service Number 41. UX Software Inc., 10 St. Mary St., Toronto, Ont. M44 1P9; (416) 964-6909. Reader Service Number 42. Vaxon Inc., 6363 Wilshire

Bvd., Ste. 127, Los Angeles, CA 90048; (213) 655-6914. Reader Service Number 43. Waterloo Microsystems, 175 W. Columbia St., Waterloo, Ont. N2L 5Z5; (519) 884-3141. Reader Service Number 44. Woodchuck Industries, 340 W. 17th St., #2B, New York, NY 10011; (212) 924-0576. Reader Service Number 45. Workspace Computer, 3848 Carson St., Ste. 301, Torrance, CA 90503; (213) 540-1553. Reader Service Number 46.

DDJ



Power Tools for power users™

Call today for our free catalog of design aids, compilers, libraries, debuggers, and support tools for Apple and IBM micro computers. The Power Tools catalog includes product descriptions, warranty and license terms, and all the information you need to make an intelligent purchase decision.

TSF offers competitive pricing, free UPS shipping on orders over \$100, and a reasonable return policy. Visa, MasterCard, and American Express accepted without surcharge. TSF helps you get your job done.

Sample Prices:

Microsoft C \$259
MASM 4.0 \$109
Turbo Pascal \$45
Mark Williams C \$375
Lets C \$59
Wendin OS Toolbox \$69
Blaise Async Manager \$137

Call Toll Free (24 hrs/7 days)
Ask For Operator 2053
1-800-543-6277
Calif: 1-800-368-7600



TSF
The Software Family™

- Dept C-1 • 649 Mission Street
- San Francisco • CA 94105
- (415) 957-0111

Circle no. 230 on reader service card.

RECOVER ERASED FILES AND REPAIR DAMAGED FILES

\$69.95

File Recovery System

Recover a file that was erased
Repair files that 'won't load'
Edit a file or anything on the disk
Context-Sensitive Help
Change File Attributes
'Wipe' Feature

Call Now, 7 Days A Week, 24 Hours A Day
(408) 395-9568



Software™

"... More powerful than the
Norton Utility Version 3.1."
—PC Magazine

15100 EL CAMINO GRANDE,
SARATOGA, CA 95070

©1986 Software Resource Group, Inc. All rights reserved.

Circle no. 267 on reader service card.

Instant-C™ The Fastest Interpreter for C

Runs your programs 50 to 500 times faster than any other C language interpreter.

Any C interpreter can save you compile and link time when developing your programs. But only **Instant-C** saves your time by running your program at compiled-code speed.

Fastest Development. A program that runs in one second when compiled with an optimizing compiler runs in two or three seconds with **Instant-C**. Other interpreters will run the same program in two minutes. Or even ten minutes. Don't trade slow compiling and linking for slow testing and debugging. Only **Instant-C** will let you edit, test, and debug at the fastest possible speeds.

Fastest Testing. **Instant-C** immediately executes any C expression, statement, or function call, and display the results. Learn C, or test your programs faster than ever before.

Fastest Debugging. **Instant-C** gives you the best source-level debugger for C. Single-step by source statement, or set any number of conditional breakpoints throughout your program. Errors always show the source statements involved. Once you find the problem, test the correction in seconds.

Fastest Programming. **Instant-C** can directly generate executable files, supports full K & R standard C, comes with complete library source, and works under PC-DOS, MS-DOS, or CP/M-86.

Instant-C gives you working, well-tested programs faster than any other programming tool. Satisfaction guaranteed, or your money back in first 31 days. **Instant-C** is \$495.

Rational
Systems, Inc.

P.O. Box 480
Natick, MA 01760
(617) 653-6194

Circle no. 145 on reader service card.

ADVERTISER INDEX

Reader Service No.	Advertiser	Page	Reader Service No.	Advertiser	Page
158	AMPRO Computers, Inc.	89	220	Metadigm, Inc.	115
221	Alycon Corp.	47	110	Micro Interface Corp.	99
277	Alycon Corp.	105	*	Micromint, Inc.	55
273	Alys Inc.	65	105	Microprocessors Unlimited	66
121	Arity Corp.	15	*	Microsoft	67
248	Ashton-Tate	125	261	Miken Optical Co.	91
242	Ashton-Tate	27	190	Mitex	43
216	Attron	18	*	Mix Software	2
250	Austin Code Works	96	128	Morgan Computing	74
159	Blaise Computing	38	271	National Instruments	60
275	Block Island Technology	91	*	Norton Utilities (The)	43
267	Brown Bag Software	127	251	Nostradamus	123
181	C Users Group	114	137	OES Systems	89
*	C Ware	58	254	Oasys	97
246	Cardinal Point	70	124	Optotech	1
226	Cauzin Systems, Inc.	4,5	192	Overland Data Inc.	67
178	Chalcedony	39	266	Peak Electronics	93
81	Cogitate, Inc.	114	76	Personal Tex	47
237	CompuServe	C2	91	Phoenix Computer Products	C4
122	CompuView	23	139	Phoenix Computer Products	113
225	Computer Publishing Society	101	193	Plu Perfect Systems	101
96	Computer Innovations	C3	169	Poor Person Software	91
82	Creative Programming	53	229	Port A Soft	70
268	Custom Software Systems	100	*	Precise Electronics	96
265	D&W Digital	16	140	Productivity Products Int'l	41
86	Daisoft System	70	143	Programmer's Shop	72,73
263	Data Management Consultants	69	141	Programmer's Shop	75
203	Datalight	67	264	QCAD System	100
258	Desktop AI	92	205	Queio	66
87	Digital Research Computers	83	107	Quilt Computing	66
204	Disclose	102	206	Raima Corp.	17
*	Allen Holub-Shell (House ad)	71	145	Rational Systems	128
*	C-Products (House ad)	117	*	SAS Institute Inc.	29
*	DDJ Bound Volume (House ad)	103	78	SLR Systems	61
*	DDJ Subscription (House ad)	104	114	Seidl Computer Engineering	91
*	Sourcebook (House ad)	111	85	Semi Disk Systems	92
*	280 Toolbook (House ad)	64	202	Simon & Schuster Computer Books	109
*	DDJ Mac Reprint (House ad)	113	219	Simon & Schuster Computer Books	54
*	DDJ Back Issues (House ad)	81	105	Soft Advances	105
179	Earth Computers	105	83	Soft Focus	109
89	Ecosoft, Inc.	13	259	SoftCraft, Inc.	20
90	Edward K. Ream	99	113	Software Horizons Inc.	112
138	Essential Software	42	262	Solution Systems	48
93	FairCom	61	152	Solution Systems	107
94	Fox Software	52	155	Solution Systems	77
*	GTE Government System	85	147	Solution Systems	107
97	Gimple Software	37	148	Solution Systems	107
132	Greenleaf Software	22	153	Solution Systems	48
274	Harvard Softworks	118	164	Spruce Technology Corp.	93
194	Hauppauge Computer Works	63	172	Sunny Hill Software	88
*	InfoPro Systems	74	175	TLM Systems	61
*	Integral Quality	89	173	TLM Systems	57
260	Interface Group (The)	87	174	TLM Systems	59
257	Logitech, Inc.	19	230	TSF	127
98	MGlobal	34	245	Tech PC	21
270	Majic Software Inc.	121	231	Terra Base Software	95
109	Manx Software	107	234	Think Technology Inc.	35
222	Manx Software	119	77	UniPress Software	31
223	Manx Software	56	112	Wendin, Inc.	9
108	Manx Software	7	116	Wizard Systems	95
102	Mark Williams	11	208	Wordtech Systems	33
189	Martian Technologies	102	244	Workman & Associates	92
227	Media Cybernetics	49	*	Worldwide Access Inc.	79

*This advertiser prefers to be contacted directly: see ad for phone number.

Advertising Sales Offices

East Coast
Walter Andrzejewski (617) 868-1524

Northern California/Northwest
Lisa Boudreau (415) 366-3600

Midwest
Michele Beaty (317) 875-8093

Southern California/AZ/NM/TX
Michael Wiener (415) 366-3600

Advertising Director
Shawn Horst (415) 366-3600

"The C86 C Compiler is Great..."

Computer Innovations' Support is Even GREATER"

DALE HILLMAN,
PRESIDENT, XOR CORPORATION
CREATOR OF "NFL CHALLENGE"



When Dale Hillman decided to create the most exciting football simulation game ever, he knew he needed good language support. The portability and maintainability of C made it a natural choice. Which C compiler to choose was another matter entirely.

"Of the many C compilers available, choosing the best one for the job was not easy. Comparing benchmarks, most compilers were strong in one or two categories, yet decidedly weak in others.

Computer Innovations' C86 was the exception. I found the C86 Compiler consistently strong in all categories.

"C86 had a reputation for being a solid, reliable, high-performance compiler. 8087 math support, source level debugging — it had it all. BEST of all was Computer Innovations' incredible technical support. Their highly knowledgeable support team was always available. Their assistance helped cut development

time substantially. And since NFL CHALLENGE took 12 1/2 man-years to create — every little bit helped. It was a service you just can't place a dollar value on..."

If you're working on the next great program, call Computer Innovations. We'll show you why you'll never have to look any further than C86.

**For Further Details
Call Toll-Free:
800-922-0169**

Behind Innovative Programs – Computer Innovations

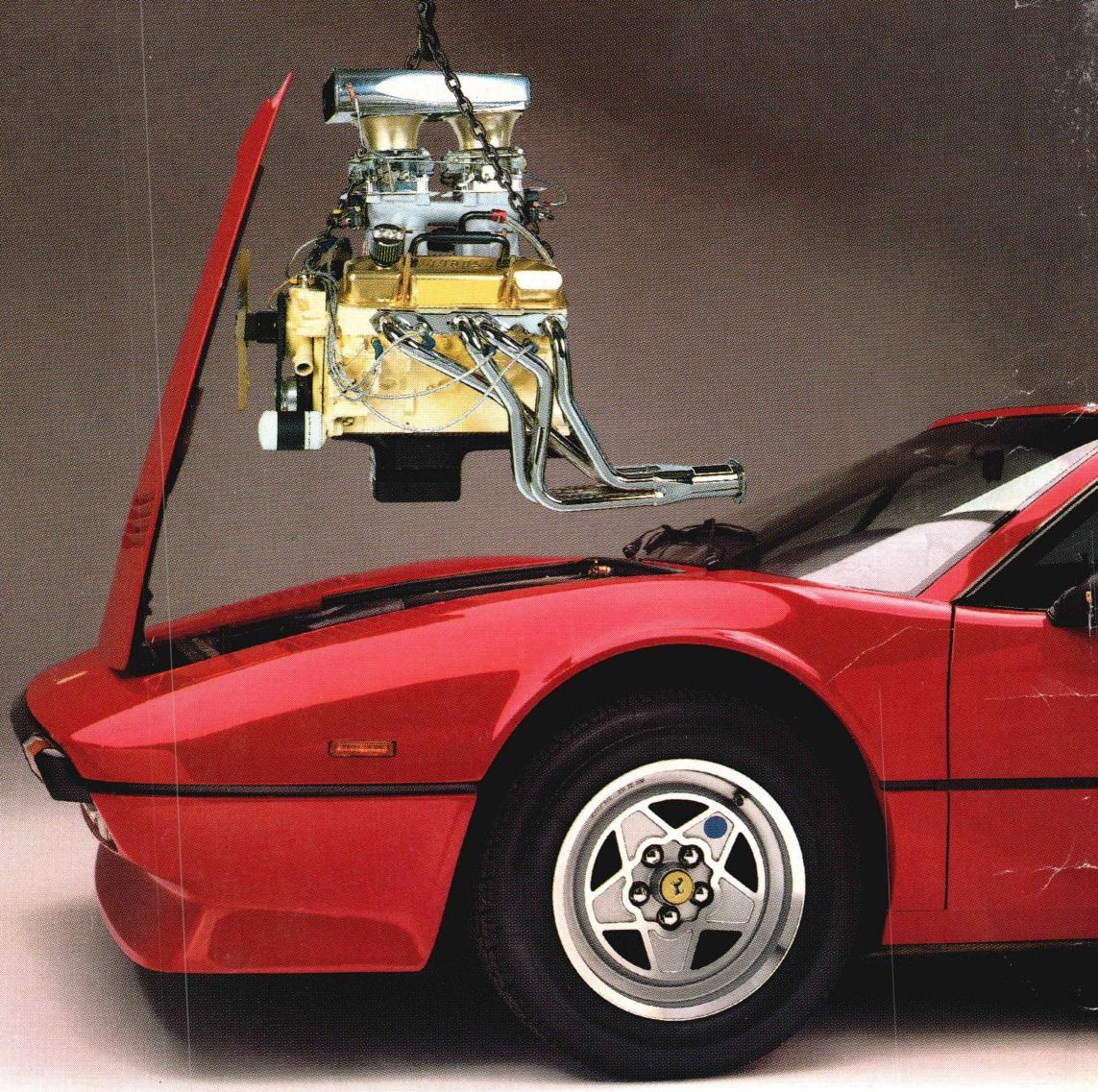


**COMPUTER
INNOVATIONS, INC.**

980 Shrewsbury Avenue,
Tinton Falls, NJ 07724 USA (201) 542-5920

EUROPEAN DISTRIBUTOR
Boston Micro, Inc., TELEX: 6712477 BMI USA

©1986 Computer Innovations, Inc.
* NFL Challenge is a trademark of NFL Properties



AT™ Pfantasies for your PC or XT™

Want better speed and memory on your PC or XT without buying an AT?

You've got it!

Phoenix's new Pfaster™286 co-processor board turns your PC or XT into a high-speed engine 60 percent faster than an AT. Three times faster than an XT. It even supports PCs with third-party hard disks. But that's only the beginning.

You can handle spreadsheets and programs you never thought possible. Set up RAM disks in both 8088 and 80286 memory for linkage editor overlays or super-high-speed disk caching. All with Pfaster286's 1mb of standard RAM, expandable to 2mb, and dual-mode design.

You can develop 8086/186/286 software on your XT faster. Execute 95 percent of the application packages that run on the AT, excluding those that require fancy I/O capabilities your PC or XT hardware just isn't designed to handle. Queue multi-copy, multi-format print jobs for spooling. Or, switch to native 8088 mode to handle



hardware-dependent programs and back again without rebooting. All with Pfaster286's compatible ROM software. And, Pfaster286 does the job unintrusively! No motherboard to exchange. No wires to solder. No chips to pull. Just plug it into a standard card slot, and type the magic word, "PFAST."

If you really didn't want an AT in the first place, just what it could do for you, call or write: Phoenix Computer Products Corp., 320 Norwood Park South, Norwood, MA 02062; (800) 344-7200. In Massachusetts, 617-762-5030.

Programmers' Pfantasies™
by

Phoenix

XT and AT are trademarks of International Business Machines Corporation. Pfaster286 and Programmers' Pfantasies are trademarks of Phoenix Computer Products Corporation. We are showing the addition of a second engine to symbolize how Pfaster can be added to your PC or XT to increase performance.

For the Ferrari aficionado: yes, we know this is a rear engine car. We are showing the addition of a second engine to symbolize how Pfaster can be added to your PC or XT to increase performance.

Circle no. 91 on reader service card.